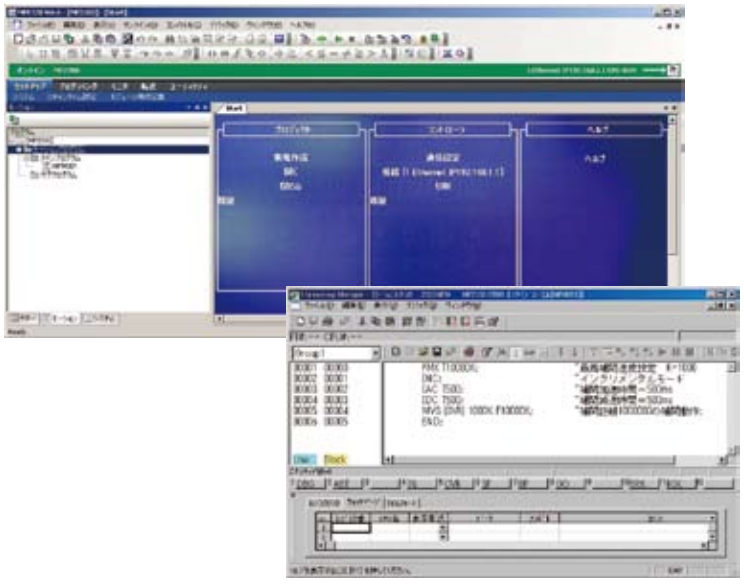
YASKAWA

Machine Controller MP2000 Series

USER'S MANUAL

for Motion Programming



Overview 1

Specifications 2

Program Development Flow 3

Motion Programs 4

Sequence Programs 5

Variables (Registers) 6

Programming 7

Command Reference 8

Engineering Tool MPE720 9

Troubleshooting 10

Appendices App

MANUAL NO. SIEP C880700 38B

Copyright © 2008 YASKAWA ELECTRIC CORPORATION

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form, or by any means, mechanical, electronic, photocopying, recording, or otherwise, without the prior written permission of Yaskawa. No patent liability is assumed with respect to the use of the information contained herein. Moreover, because Yaskawa is constantly striving to improve its high-quality products, the information contained in this manual is subject to change without notice. Every precaution has been taken in the preparation of this manual. Nevertheless, Yaskawa assumes no responsibility for errors or omissions. Neither is any liability assumed for damages resulting from the use of the information contained in this publication.

About This Manual

- This manual provides information on motion commands for the MP2000 series Machine Controllers.
  - Motion program overview
  - Specifications
  - Program development flow
  - Motion programs and sequence programs
  - Variables
  - Programming
  - Command reference
  - Engineering tool MPE720
  - Troubleshooting, etc.
- Read this manual carefully to ensure the proper use of the MP2000 series Machine Controller. Also, keep this manual in a safe place so that it can be referred to whenever necessary.

Using This Manual

- Intended Audience

  This manual is intended for the following users.
  - Those responsible for designing the MP2000 series Machine Controller system
  - Those responsible for writing MP2000 series Machine Controller motion programs and sequence programs
- Engineering Tool MPE720 Version Number

  In this manual, the operation of MPE720 is described using screenshots of MPE720 version 6.

  For this reason, the screenshots and some descriptions may differ from those for MPE720 version 5.
- Description of Abbreviation

  In this manual, the following abbreviation is used.
  - MP2000: Machine controller model including MP2100, MP2100M, MP2200, MP2300, MP2300S, MP2310, MP2400, MP2500, MP2500M, MP2500D, and MP2500MD

Manuals for MP2000 Series

- The user's manuals are prepared by classifying MP2000 series Machine Controller models into MP2100, MP2100M, MP2200, MP2300, MP2300S, MP2310, MP2400, MP2500, MP2500M, MP2500D, and MP2500MD.
  Refer to Related Manuals on the next page as required.

Related Manuals

- The following table lists the related manuals. Refer to these manuals as required.
- Before using, be sure you understand the product conditions, including specifications and usage restrictions.

Manual Name	Manual Number	Contents
Machine Controller MP2100/MP2100M User's Manual	SIEPC88070001	Describes the functions, specifications, setup procedures, and operating methods of the MP2100/MP2100M.
Machine Controller MP2200 User's Manual	SIEPC88070014	Describes the functions, specifications, setup procedures, and operating methods of the MP2200.
Machine Controller MP2300 Basic Module User's Manual	SIEPC88070003	Describes the functions, specifications, setup procedures, and operating methods of the MP2300.
Machine Controller MP2300S Basic Module User's Manual	SIEPC88073200	Describes the functions, specifications, setup procedures, and operating methods of the MP2300S.
Machine Controller MP2310 Basic Module User's Manual	SIEPC88073201	Describes the functions, specifications, setup procedures, and operating methods of the MP2310.
Machine Controller MP2400 User's Manual	SIJPC88074200	Describes the functions, specifications, setup procedures, and operating methods of the MP2400.
Machine Controller MP2500/MP2500M/MP2500D/ MP2500MD User's Manual	SIEPC88075200	Describes the functions, specifications, setup procedures, and operating methods of the MP2500/MP2500M/MP2500D/MP2500MD.
Machine Controller MP2000 series SVB/SVB-01 Motion Module User's Manual	SIEPC88070033	Describes the functions, specifications, and application methods of the MP2000-series Motion Module that is built into the SVB and SVB-01 Module.
Machine Controller MP2000 Series Motion Module SVA-01 User's Manual	SIEPC88070032	Describes the functions, specifications, and operating methods of MP2000-series Motion Module SVA-01.
Machine Controller MP2000 Series Pulse Output Motion Module PO-01 User's Manual	SIEPC88070028	Describes the functions, specifications, and operating methods of MP2000-series Motion Module PO-01.
Machine Controller MP2000 Series Communication Module User's Manual	SIEPC88070004	Describes the functions, specifications, and application methods of the MP2000 series Communication Modules.
Engineering Tool for Machine Controller MP2000 Series MPE720 Version 6 User's Manual	SIEPC88070030	Describes the installation and operation of the programming software MPE720 for MP2000 series.
Machine Controller MP900/MP2000 Series MPE720 Software for Programming Device User's Manual	SIEPC88070005	Describes the installation and operation of the programming software MPE720 for MP900/MP2000 series.
Machine Controller MP900/MP2000 Series User's Manual, Ladder Programming	SIEZ-C887-1.2	Describes the processing instructions used in MP900/MP2000 series Machine Controller ladder programs.
Machine Controller MP900/MP2000 Series New Ladder Editor User's Manual Programming Manual	SIEZ-C887-13.1	Describes the programming instructions of the New Ladder Editor, which assists MP900/MP2000-series design and maintenance.
Machine Controller MP900/MP2000 Series New Ladder Editor User's Manual Operation	SIEZ-C887-13.2	Describes the operating methods of the New Ladder Editor, which assists MP900/ MP2000-series design and maintenance.

Visual Aids

The following aids are used to indicate certain types of information for easier reference.

IMPORTANT — Indicates important information that should be memorized, including precautions such as alarm displays to avoid damaging the devices.

◀ EXAMPLE ▶ — Indicates supplemental information.

INFO — Indicates application examples.

TERMS — Indicates definitions of difficult terms or terms that have not been previously explained in this manual.

Safety Information

The following conventions are used to indicate precautions in this manual. Information marked as shown below is important for the safety of the user. Always read this information and heed the precautions that are provided. The conventions are as follows:

⚠ WARNING — Indicates precautions that, if not heeded, could possibly result in loss of life or serious injury.

⚠ CAUTION — Indicates precautions that, if not heeded, could result in relatively serious or minor injury, or property damage.

If not heeded, even precautions classified under ⚠ CAUTION can lead to serious results depending on circumstances.

⃠ PROHIBITED — Indicates prohibited actions. Specific prohibitions are indicated inside ⃠.

For example,  indicates no fire or open flame.

MANDATORY — Indicates mandatory actions. Specific actions are indicated inside ●.

For example,  indicates that grounding is required.

Safety Precautions

This section describes important precautions that apply to motion programming. Before programming, always read this manual and all other attached documents to ensure correct programming.
Before using the equipment, familiarize yourself with equipment details, safety information, and all other precautions.

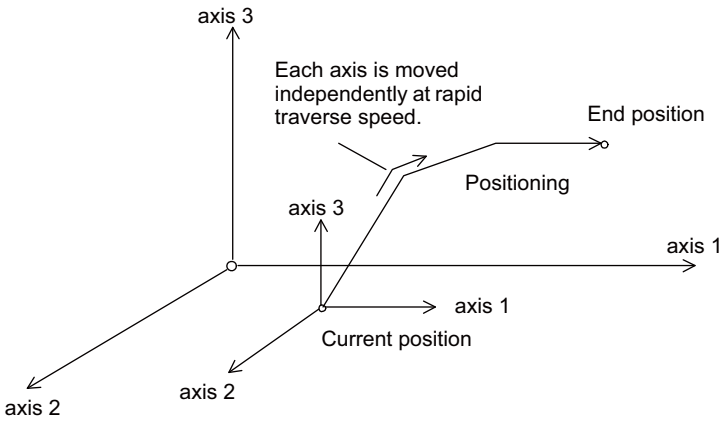
■ Application Precautions

⚠ CAUTION

- When programming the following axis move commands, check the path to make sure that there are no tools or other obstacles in the way of the workpiece.

  The axis move commands that must be checked are as follows:
  - Positioning (MOV)
  - Linear Interpolation (MVS)
  - Circular Interpolation (MCC, MCW)
  - Helical Interpolation (MCC, MCW)
  - Set Time Positioning (MVT)
  - Linear Interpolation with Skip Function (SKP)
  - Zero Point Return (ZRN)
  - External Positioning (EXM)

◀ EXAMPLE ▶



Example of Basic Path for Positioning (MOV)

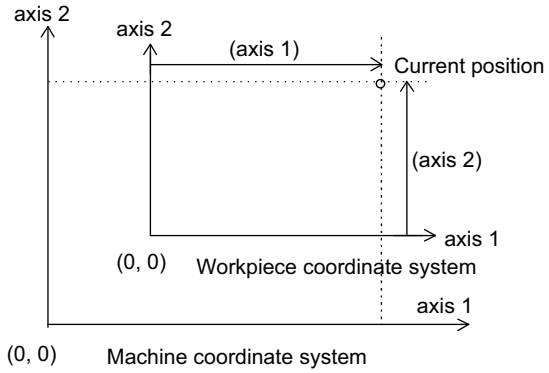
Failure to carry out the above checks may result in damage to equipment, serious personal injury, or even death.

⚠ CAUTION

- If the following coordinate commands are designated incorrectly, the subsequent move operations will be entirely different than those expected. Before starting operations, be sure to check that the settings are designated correctly.

  The coordinate commands that must be checked are as follows:
  - Absolute Programming Mode (ABS)
  - Incremental Programming Mode (INC)
  - Current Position Set (POS)
  - Move ON Machine Coordinates (MVM)

  ◀ EXAMPLE ▶



Example of Work Coordinate System Created with Current Position Set (POS)

Failure to carry out the above checks may result in damage to equipment, serious personal injury, or even death.

■ General Precautions

Observe the following general precautions to ensure safe application.

- MP2000-series Machine Controller was not designed or manufactured for use in devices or systems directly related to human life. Users who intend to use the product described in this manual for special purposes such as devices or systems relating to transportation, medical, space aviation, atomic power control, or underwater use must contact Yaskawa Electric Corporation beforehand.
- MP2000-series Machine Controller has been manufactured under strict quality control guidelines. However, if this product is to be installed in any location in which a failure of MP2000-series Machine Controller involves a life and death situation or in a facility where failure may cause a serious accident, safety devices MUST be installed to minimize the likelihood of any accident.
- Drawings and photos in this manual show typical product examples that may differ somewhat from the product delivered.
- We will update the data sheet number for the manual and issue revisions when changes are made. The edition number of the revised manual appears on the back of the manual.
- Contact your Yaskawa representative and quote the data sheet number on the front page of the manual if you need to replace a manual that was lost or destroyed.
- Contact your Yaskawa representative to order new nameplates whenever a nameplate becomes worn or damaged.

Warranty

(1) Details of Warranty

■ Warranty Period

The warranty period for a product that was purchased (hereinafter called “delivered product”) is one year from the time of delivery to the location specified by the customer or 18 months from the time of shipment from the Yaskawa factory, whichever is sooner.

■ Warranty Scope

Yaskawa shall replace or repair a defective product free of charge if a defect attributable to Yaskawa occurs during the warranty period above. This warranty does not cover defects caused by the delivered product reaching the end of its service life and replacement of parts that require replacement or that have a limited service life.

This warranty does not cover failures that result from any of the following causes.

1. Improper handling, abuse, or use in unsuitable conditions or in environments not described in product catalogs or manuals, or in any separately agreed-upon specifications
2. Causes not attributable to the delivered product itself
3. Modifications or repairs not performed by Yaskawa
4. Abuse of the delivered product in a manner in which it was not originally intended
5. Causes that were not foreseeable with the scientific and technological understanding at the time of shipment from Yaskawa
6. Events for which Yaskawa is not responsible, such as natural or human-made disasters

(2) Limitations of Liability

1. Yaskawa shall in no event be responsible for any damage or loss of opportunity to the customer that arises due to failure of the delivered product.
2. Yaskawa shall not be responsible for any programs (including parameter settings) or the results of program execution of the programs provided by the user or by a third party for use with programmable Yaskawa products.
3. The information described in product catalogs or manuals is provided for the purpose of the customer purchasing the appropriate product for the intended application. The use thereof does not guarantee that there are no infringements of intellectual property rights or other proprietary rights of Yaskawa or third parties, nor does it construe a license.
4. Yaskawa shall not be responsible for any damage arising from infringements of intellectual property rights or other proprietary rights of third parties as a result of using the information described in catalogs or manuals.

(3) Suitability for Use

1. It is the customer's responsibility to confirm conformity with any standards, codes, or regulations that apply if the Yaskawa product is used in combination with any other products.
2. The customer must confirm that the Yaskawa product is suitable for the systems, machines, and equipment used by the customer.
3. Consult with Yaskawa to determine whether use in the following applications is acceptable. If use in the application is acceptable, use the product with extra allowance in ratings and specifications, and provide safety measures to minimize hazards in the event of failure.
   - Outdoor use, use involving potential chemical contamination or electrical interference, or use in conditions or environments not described in product catalogs or manuals
   - Nuclear energy control systems, combustion systems, railroad systems, aviation systems, vehicle systems, medical equipment, amusement machines, and installations subject to separate industry or government regulations
   - Systems, machines, and equipment that may present a risk to life or property
   - Systems that require a high degree of reliability, such as systems that supply gas, water, or electricity, or systems that operate continuously 24 hours a day
   - Other systems that require a similar high degree of safety
4. Never use the product for an application involving serious risk to life or property without first ensuring that the system is designed to secure the required level of safety with risk warnings and redundancy, and that the Yaskawa product is properly rated and installed.
5. The circuit examples and other application examples described in product catalogs and manuals are for reference. Check the functionality and safety of the actual devices and equipment to be used before using the product.
6. Read and understand all use prohibitions and precautions, and operate the Yaskawa product correctly to prevent accidental harm to third parties.

(4) Specifications Change

The names, specifications, appearance, and accessories of products in product catalogs and manuals may be changed at any time based on improvements and other reasons. The next editions of the revised catalogs or manuals will be published with updated code numbers. Consult with your Yaskawa representative to confirm the actual specifications before purchasing a product.

Contents

About This Manual ---- iii
Using This Manual ---- iii
Manuals for MP2000 Series ---- iii
Related Manuals ---- iv
Visual Aids ---- v
Safety Information ---- v
Safety Precautions ---- vi
Warranty ---- viii

1 Overview

1.1 What is a Motion Program? ---- 1-2
1.2 Motion Program Features ---- 1-3
1.2.1 Execution Method ---- 1-3
1.2.2 Motion Control in Full Synchronization with Sequence Control ---- 1-3
1.2.3 Easy to Realize High-level Motion Control ---- 1-4
1.2.4 Easy-to-Understand Motion Language ---- 1-4
1.2.5 Arithmetic Operations ---- 1-4
1.2.6 Data Transfer from/to Ladder Program ---- 1-5
1.2.7 Memory Usage Reduced by Use of Subprograms ---- 1-5
1.2.8 Parallel Program Execution ---- 1-6
1.2.9 Program Online Editing ---- 1-6
1.2.10 Enriched Easy Programming Functions (MPE720 Ver.6.04 or later) ---- 1-7
1.3 Motion Program Execution Sequence ---- 1-8
1.4 Motion Program Execution Registration ---- 1-9
1.5 Motion Program Execution Timing ---- 1-10
1.6 Grouping ---- 1-11
1.7 Application Examples ---- 1-12
1.7.1 Example 1: Handling System ---- 1-12
1.7.2 Example 2: Mechanical Parts Inserting Machine ---- 1-12
1.7.3 Example 3: Panel Processing Machine ---- 1-13
1.7.4 Example 4: Metal Sheet Bending Equipment ---- 1-13
1.8 What is a Sequence Program? ---- 1-14
1.9 Sequence Program Features ---- 1-15
1.9.1 Execution Method ---- 1-15
1.9.2 Programming Language Commonly Used in Motion Programs ---- 1-15
1.9.3 Data Transfer from/to Motion Program ---- 1-15
1.9.4 Memory Usage Reduced by Use of Subprograms ---- 1-16
1.9.5 Easy Programming Functions (MPE720 Ver.6.04 or later) ---- 1-16

2 Specifications

2.1 MP2000 Series Machine Controller Specifications - 2-2
2.1.1 Applicable Machine Controller Models - - - - - - - - - - - - - - - - - - -

5 Sequence Programs

5.1 Sequence Program Types - - -

8 Command Reference

8.1 Axis Setting Commands ---------- 8-3
8.1.1 Absolute Mode (ABS) ---------- 8-3
8.1.2 Incremental Mode (INC) ---------- 8-7
8.1.3 Acceleration Time Change (ACC) ---------- 8-11
8.1.4 Deceleration Time Change (DCC) ---------- 8-17
8.1.5 S-curve Time Constant Change (SCC) ---------- 8-23
8.1.6 Set Velocity (VEL) ---------- 8-29
8.1.7 Maximum Interpolation Feed Speed Setting (FMX) ---------- 8-35
8.1.8 Interpolation Feed Speed Ratio Setting (IFP) ---------- 8-37
8.1.9 Interpolation Acceleration Time Change (IAC) ---------- 8-40
8.1.10 Interpolation Deceleration Time Change (IDC) ---------- 8-43
8.2 Axis Move Commands ---------- 8-46
8.2.1 Positioning (MOV) ---------- 8-46
8.2.2 Linear Interpolation (MVS) ---------- 8-50
8.2.3 Clockwise/Counterclockwise Circular Interpolation with Center Position Designation (MCW, MCC) ---------- 8-55
8.2.4 Clockwise/Counterclockwise Circular Interpolation with Radius Designation (MCW, MCC) ---------- 8-61
8.2.5 Clockwise/Counterclockwise Helical Interpolation with Center Position Designation (MCW, MCC) ---------- 8-65
8.2.6 Clockwise/Counterclockwise Helical Interpolation with Radius Designation (MCW, MCC) ---------- 8-68
8.2.7 Zero Point Return (ZRN) ---------- 8-70
8.2.8 Linear Interpolation with Skip Function (SKP) ---------- 8-72
8.2.9 Set Time Positioning (MVT) ---------- 8-74
8.2.10 External Positioning (EXM) ---------- 8-76
8.3 Axis Control Commands ---------- 8-78
8.3.1 Current Position Set (POS) ---------- 8-78
8.3.2 Move On Machine Coordinates (MVM) ---------- 8-80
8.3.3 Program Current Position Update (PLD) ---------- 8-81
8.3.4 In-Position Check (PFN) ---------- 8-82
8.3.5 Set In-Position Range (INP) ---------- 8-84
8.3.6 Coordinate Plane Setting (PLN) ---------- 8-86
8.4 Program Control Commands ---------- 8-87
8.4.1 Branching Commands (IF ELSE IEND) ---------- 8-87
8.4.2 Repeat (WHILE WEND) ---------- 8-89
8.4.3 Parallel Execution (PFORK, JOINTO, PJOINT) ---------- 8-92
8.4.4 Selective Execution (SFORK, JOINTO, SJOINT) ---------- 8-95
8.4.5 Motion Subprogram Call (MSEE) ---------- 8-99
8.4.6 Sequence Subprogram Call (SSEE) ---------- 8-100
8.4.7 User Function Call From Motion Program (UFC) ---------- 8-101
8.4.8 User Function Call from Sequence Program (FUNC) ---------- 8-109
8.4.9 Program End (END) ---------- 8-110
8.4.10 Subprogram End (RET) ---------- 8-111
8.4.11 Dwell Time (TIM) ---------- 8-112
8.4.12 I/O Variable Wait (IOW) ---------- 8-113
8.4.13 One Scan Wait (EOX) ---------- 8-115

8.4.14 Single-block Signal Disabled (SNGD)/Single-block Signal Enabled (SNGE) - - - - - - - - - - 8-116
8.5 Arithmetic Operations - 8-117
8.5.1 Substitute (=) - 8-117
8.5.2 Add (+) - 8-118
8.5.3 Subtract (-) - 8-119
8.5.4 Multiply (*) - 8-120
8.5.5 Divide (/) - 8-121
8.5.6 Remainder (MOD) - 8-122
8.6 Logic Operation - 8-123
8.6.1 OR (|) - 8-123
8.6.2 AND (&) - 8-125
8.6.3 XOR (^) - 8-126
8.6.4 NOT (!) - 8-127
8.7 Data Comparisons - 8-128
8.7.1 Data Comparison Commands (==, <>, >, <, >=, <=) - 8-128
8.8 Data Operations - 8-130
8.8.1 Bit Right Shift (SFR) - 8-130
8.8.2 Bit Left Shift (SFL) - 8-131
8.8.3 Block Move (BLK) - 8-132
8.8.4 Clear (CLR) - 8-133
8.8.5 ASCII Conversion 1 (ASCII) - 8-134
8.9 Basic Functions - 8-136
8.9.1 Sine (SIN) - 8-136
8.9.2 Cosine (COS) - 8-138
8.9.3 Tangent (TAN) - 8-139
8.9.4 Arc Sine (ASN) - 8-140
8.9.5 Arc Cosine (ACS) - 8-141
8.9.6 Arc Tangent (ATN) - 8-142
8.9.7 Square Root (SQT) - 8-143
8.9.8 BCD to Binary (BIN) - 8-145
8.9.9 Binary to BCD (BCD) - 8-146
8.9.10 Set Bit (S{ }) - 8-147
8.9.11 Reset Bit (R{ }) - 8-148
8.9.12 Rising Pulse (PON) - 8-149
8.9.13 Falling Pulse (NON) - 8-151
8.9.14 ON-Delay Timer (TON): Counting unit: 0.01 second - 8-154
8.9.15 OFF-Delay Timer (TOF):Counting unit: 0.01 second - 8-156
8.10 C-Language Control Commands - 8-158
8.10.1 C-Language Task Control (CTSK) - 8-158
8.10.2 C-Language Function Call (CFUNC) - 8-160

9 Engineering Tool MPE720

9.1 Motion Editor - - -

Appendices

A Motion Language Commands ---------- A-2
A.1 Axis Setting Commands ---------- A-2
A.2 Axis Move Commands ---------- A-3
A.3 Control Commands ---------- A-5
A.4 Program Control Commands ---------- A-6
A.5 Arithmetic Operations ---------- A-8
A.6 Logical Operations ---------- A-8
A.7 Data Comparison ---------- A-9
A.8 Data Operations ---------- A-9
A.9 Basic Functions ---------- A-10
A.10 C-Language Control Commands ---------- A-11
B Sample Programs ---------- A-12
B.1 Programs for Controlling Motion Program Execution ---------- A-13
B.2 Parallel Processing ---------- A-15
B.3 Motion Program for Speed Control ---------- A-16
B.4 Simple Synchronized Operation Using a Virtual Axis ---------- A-17
B.5 Sequence Programs ---------- A-19
C Differences between MP900 Series and MP2000 Series Machine Controllers ---------- A-21
C.1 Motion Programs ---------- A-21
C.2 Sequence Programs ---------- A-21
C.3 Motion Programming Commands ---------- A-22
C.4 Group Definitions ---------- A-22
C.5 Debug Function ---------- A-23
C.6 Motion Program Alarms ---------- A-23
D Precautions ---------- A-24
D.1 General Precautions ---------- A-24
D.2 Precautions on Motion Parameter Settings ---------- A-24

Index

Revision History

1

Overview

This chapter introduces motion programs and describes their features for those who are unfamilier with them.

1.1 What is a Motion Program? ---- 1-2

1.2 Motion Program Features ---- 1-3

1.2.1 Execution Method ---- 1-3
1.2.2 Motion Control in Full Synchronization with Sequence Control ---- 1-3
1.2.3 Easy to Realize High-level Motion Control ---- 1-4
1.2.4 Easy-to-Understand Motion Language ---- 1-4
1.2.5 Arithmetic Operations ---- 1-4
1.2.6 Data Transfer from/to Ladder Program ---- 1-5
1.2.7 Memory Usage Reduced by Use of Subprograms ---- 1-5
1.2.8 Parallel Program Execution ---- 1-6
1.2.9 Program Online Editing ---- 1-6
1.2.10 Enriched Easy Programming Functions (MPE720 Ver.6.04 or later) ---- 1-7

1.3 Motion Program Execution Sequence ---- 1-8

1.4 Motion Program Execution Registration ---- 1-9

1.5 Motion Program Execution Timing ---- 1-10

1.6 Grouping ---- 1-11

1.7 Application Examples ---- 1-12

1.7.1 Example 1: Handling System ---- 1-12
1.7.2 Example 2: Mechanical Parts Inserting Machine ---- 1-12
1.7.3 Example 3: Panel Processing Machine ---- 1-13
1.7.4 Example 4: Metal Sheet Bending Equipment ---- 1-13

1.8 What is a Sequence Program? ---- 1-14

1.9 Sequence Program Features ---- 1-15

1.9.1 Execution Method ---- 1-15
1.9.2 Programming Language Commonly Used in Motion Programs ---- 1-15
1.9.3 Data Transfer from/to Motion Program ---- 1-15
1.9.4 Memory Usage Reduced by Use of Subprograms ---- 1-16
1.9.5 Easy Programming Functions (MPE720 Ver.6.04 or later) ---- 1-16

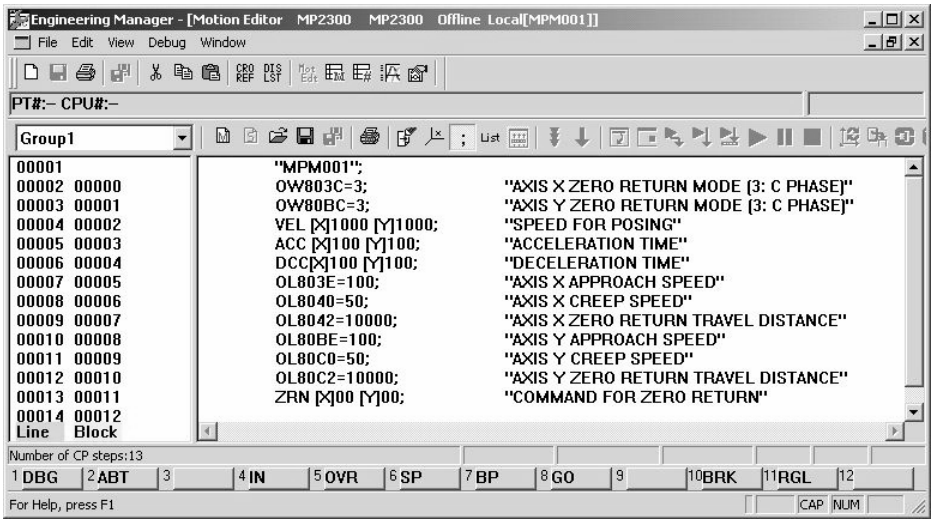
1.1 What is a Motion Program?

The motion program is a program written in motion language, Yaskawa's unique textual language.
A motion program can be executed either by coding an MSEE command in a ladder program or by registering the motion program in the program execution registry screen dialog box for the M-EXECUTOR Module.

Note: M-EXECUTOR Module compatible MP2000-series Machine Controllers are limited to the MP2100, MP2100M, MP2300S, MP2310, and MP2400. The M-EXECUTOR Module cannot be used with other MP2000-series Machine Controller models.

Apart from ladder programs, up to 256 motion programs can be created.

An example of a motion program is shown below.

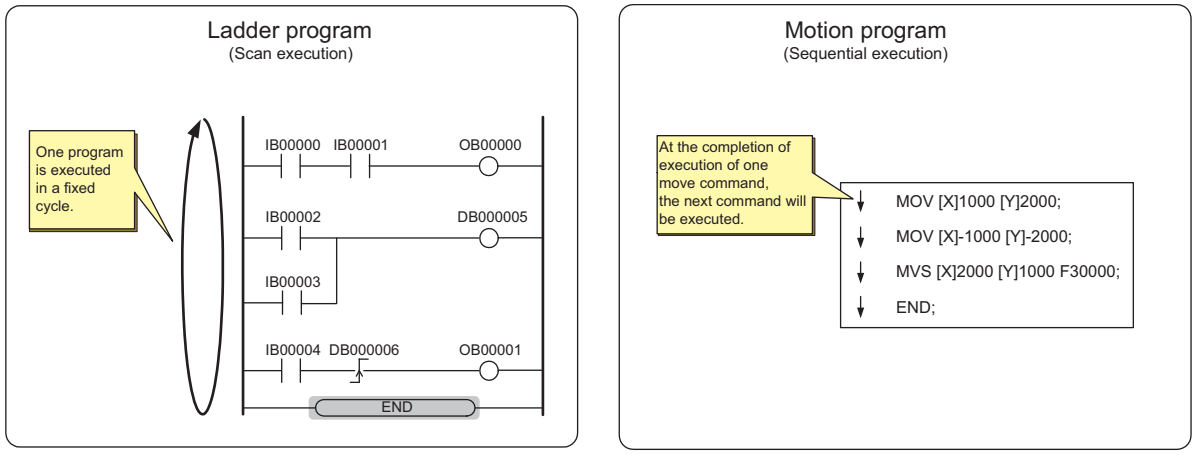


The features of motion programs are described starting from the next page.

1.2 Motion Program Features

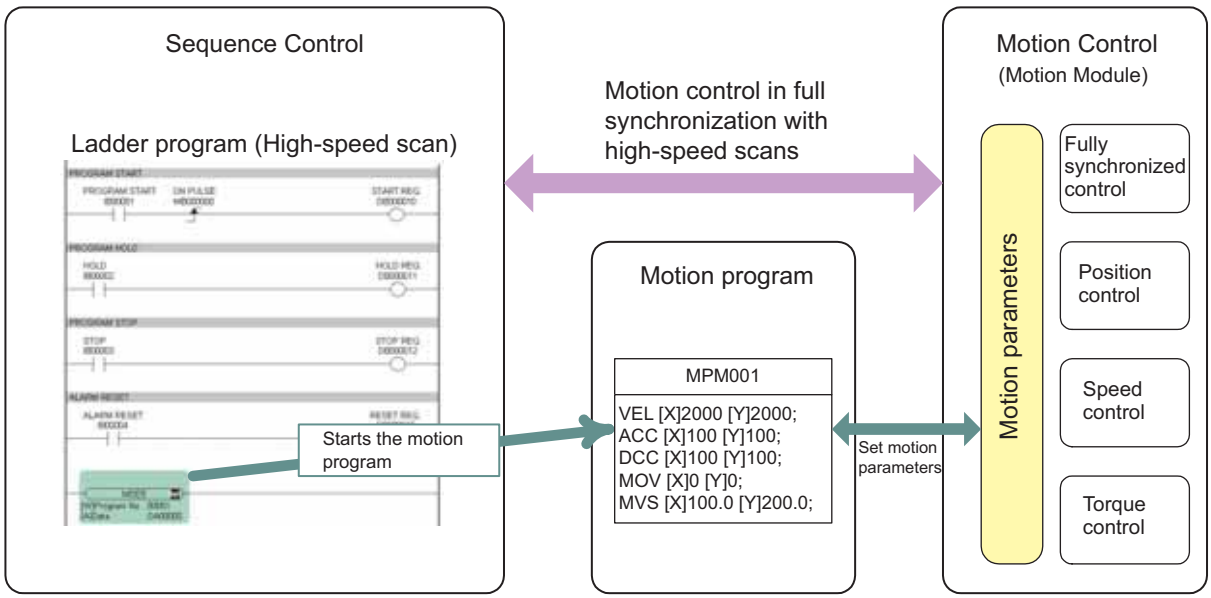
1.2.1 Execution Method

A motion program employs an execution method that differs from the ladder program.
With a ladder program, processing from the program start to an END command is completed within one scan.
With a motion program, the processing requested by one command normally requires more than one scan.
Also, the commands are executed sequentially, in the order they are coded.
In this manual, the execution method of ladder program is referred to as Scan Execution, and that of a motion program as Sequential Execution.



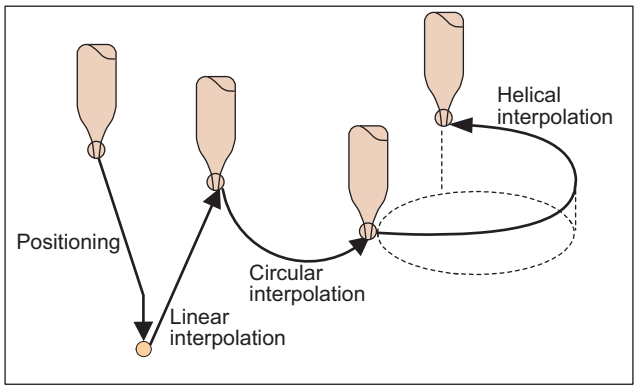
1.2.2 Motion Control in Full Synchronization with Sequence Control

The process written in a motion program is executed in full synchronization with high-speed scans of the MP2000-series Machine Controller. The axis movement will start within one scan after the start request from the ladder program, without any time lag to start the motion program.



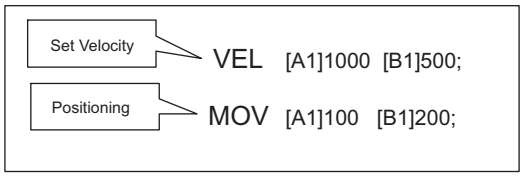
1.2.3 Easy to Realize High-level Motion Control

In addition to basic motion control, motion control that involves complicated movements can be easily realized by using motion programs.



1.2.4 Easy-to-Understand Motion Language

A motion program employs intuitive motion language commands such as VEL to set a velocity and MOV for positioning.



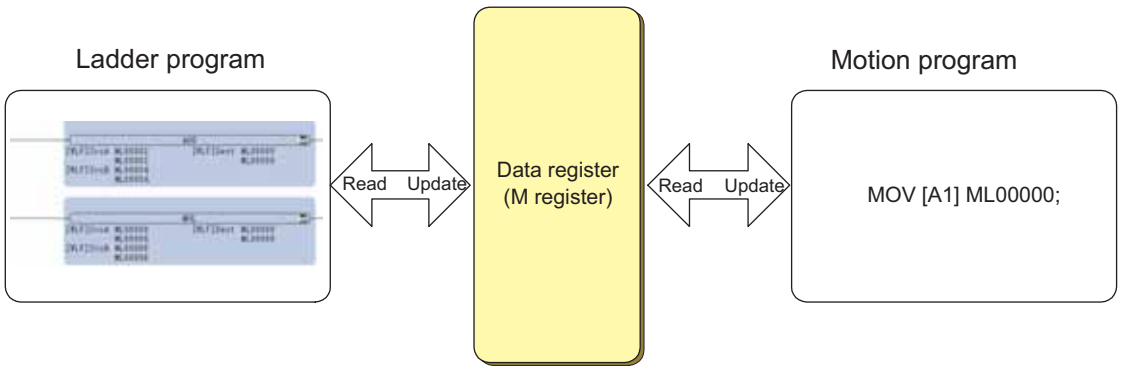
1.2.5 Arithmetic Operations

The motion language includes commands for arithmetic operations and logical operations.
These commands allow you to include various calculations, such as calculation of target position in a motion program.

```
DL00000 = DL00002 + DW00004;
DL00000 = DW00002 * DL00004;
MW00000 = MW00000 & 00FFH;
MF00000 = SIN(30.0);
```

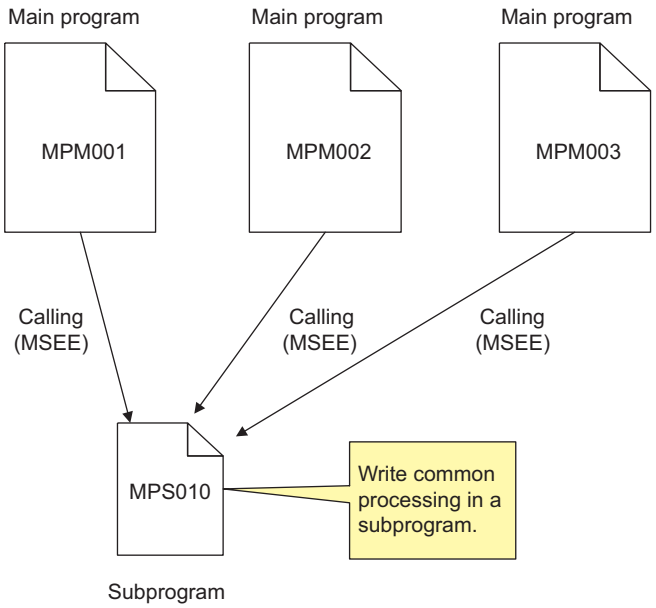
1.2.6 Data Transfer from/to Ladder Program

Data can be transferred between a ladder program and motion program.
Data registers (M registers) are used to transfer the data.
In this way, a value updated in a ladder program can be used in a motion program, and vice versa



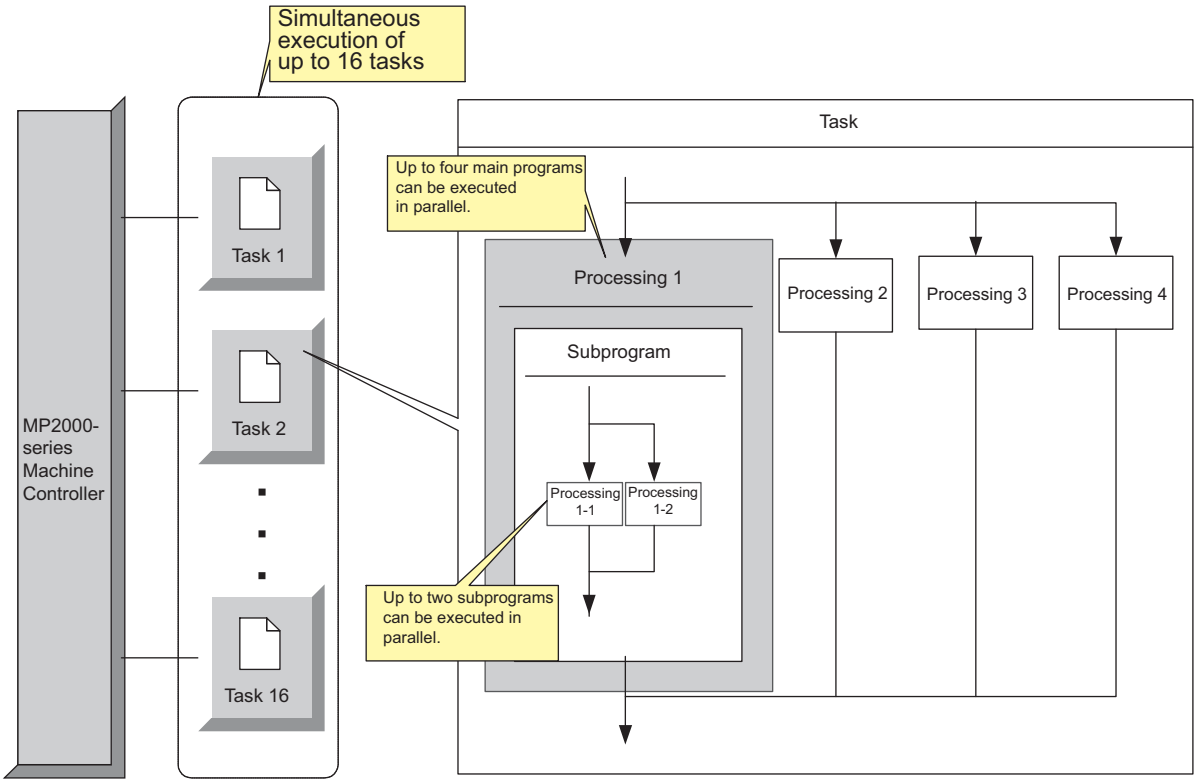
1.2.7 Memory Usage Reduced by Use of Subprograms

Subroutines (subprograms) can be created within a motion program.
The number of program steps can be minimized by creating a subprogram that includes a set of commands to perform a repeated or regular task, thus reducing memory usage.



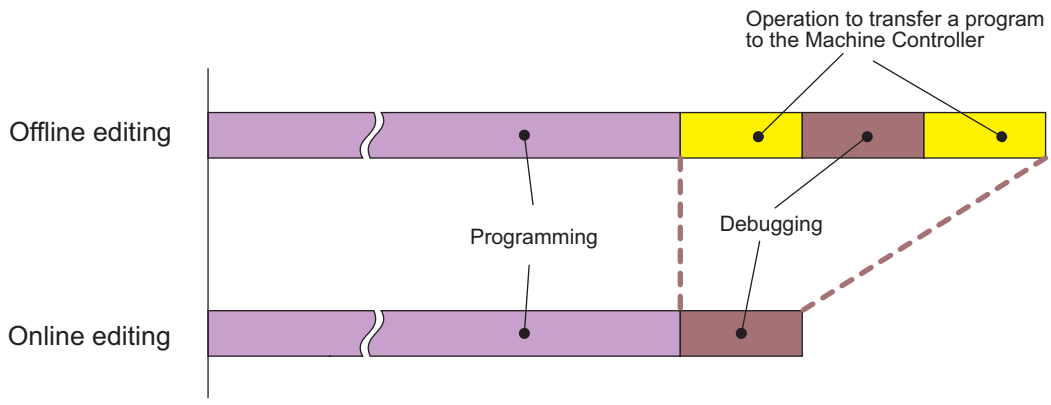
1.2.8 Parallel Program Execution

With a single MP2000-series Machine Controller, up to 16 tasks can be simultaneously executed using motion programs. With one motion program, up to four main programs can be simultaneously executed. Additionally, up to two subprograms can be simultaneously executed by calling subprograms from the main program. Multiple different movements can be simultaneously controlled by using such the parallel program execution function.



1.2.9 Program Online Editing

Motion programs can be edited online in the same way as ladder programs.
Online editing refers to editing programs with the programming device logged on to the Machine Controller. In online editing mode, the operation to save the edited program automatically transfers the saved program to the Machine Controller. Thus, an operation to transfer to the Machine Controller is not required and program development efficiency is improved.



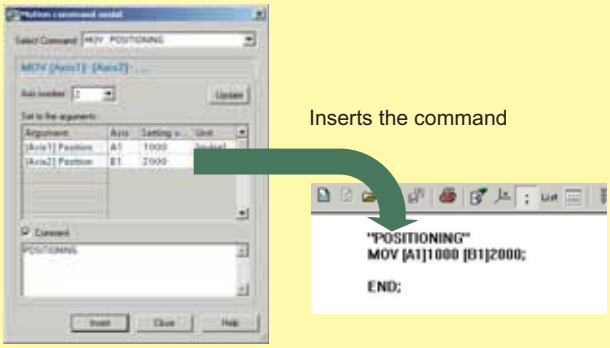
INFO Online editing is disabled while a motion program is running.

1.2.10 Enriched Easy Programming Functions (MPE720 Ver.6.04 or later)

The engineering tool MPE720 Ver.6 for MP2000-series Machine Controllers is provided with the following easy programming functions.

Command Input Assistant Function

Simply select a command and set data in the **Motion Command Assist** dialog box to insert the command in the editor.



Test Run Function

Control the axes onscreen.



Axis Operation Monitor Function

View the motion status of each axis onscreen.



Program Execution Registration Function

Easily register programs to be executed in the system.



Operation Control Panel Function

Start motion programs from the **Motion Editor** window.



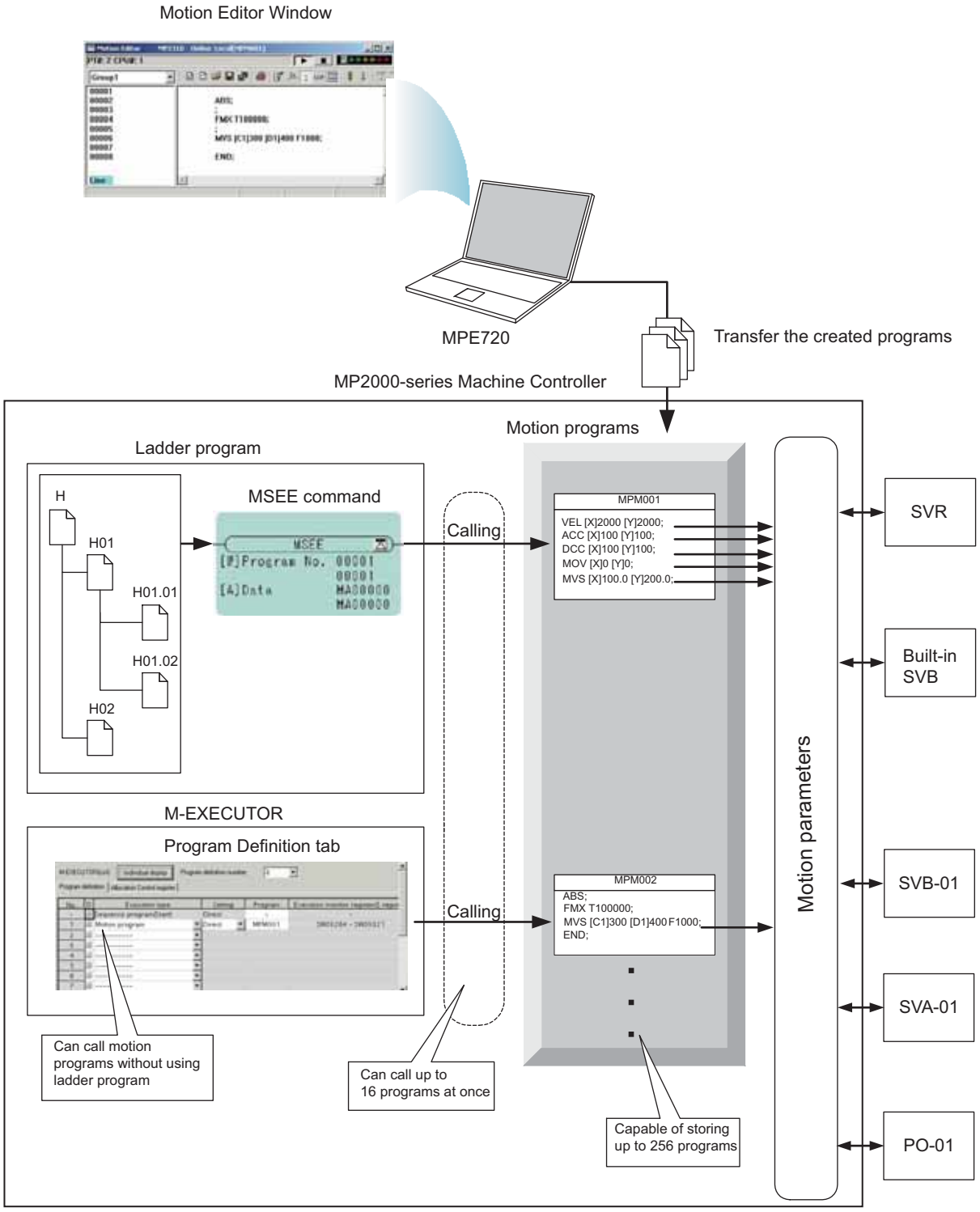
Debug Function

Debug a motion program.
The debug commands, including step-by-step execution and break point setting, are provided.



1.3 Motion Program Execution Sequence

The motion programs created on the MPE720 **Motion Editor** window are transferred to the MP2000-series Machine Controller. The transferred motion programs can be called by MSEE commands coded in the ladder program, or from the execution registry screen dialog box of the M-EXECUTOR Module. Motion commands are sent to the motion module via the motion parameters to move axes.
The following diagram illustrates how motion programs created using the MPE720 are executed.

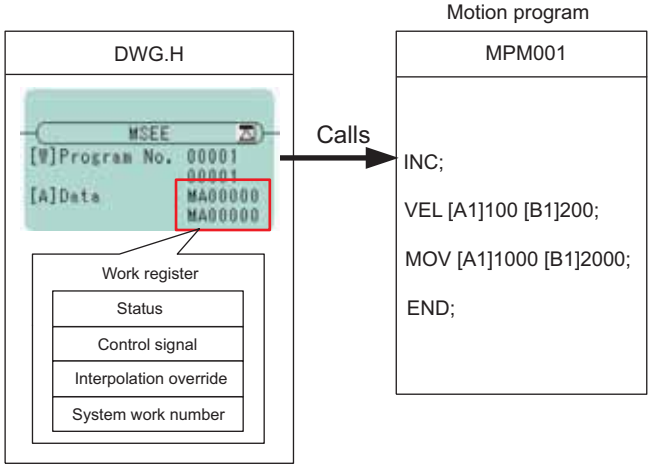


1.4 Motion Program Execution Registration

Execution of motion programs can be registered in two ways.

■ Calling a Motion Program from the Ladder Program

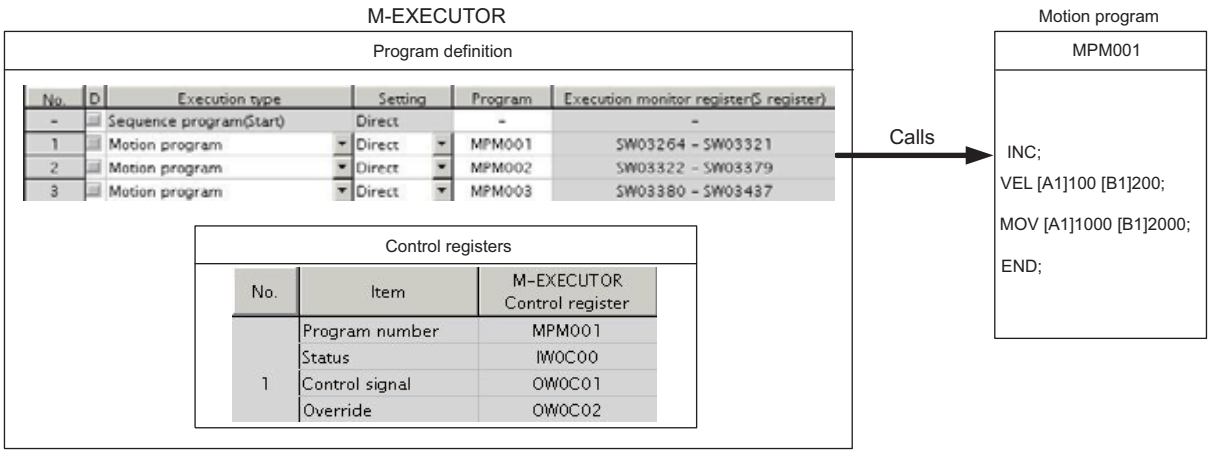
Code an MSEE command in an H drawing to call the motion program to run. An MSEE work register is used to start/stop the called motion program. Motion programs can be called from any H drawing: parent drawing, child drawing, or grandchild drawing.



INFO In this manual, the high-speed processing drawing of a ladder program is referred to as H drawing.

■ Registering Motion Programs in M-EXECUTOR

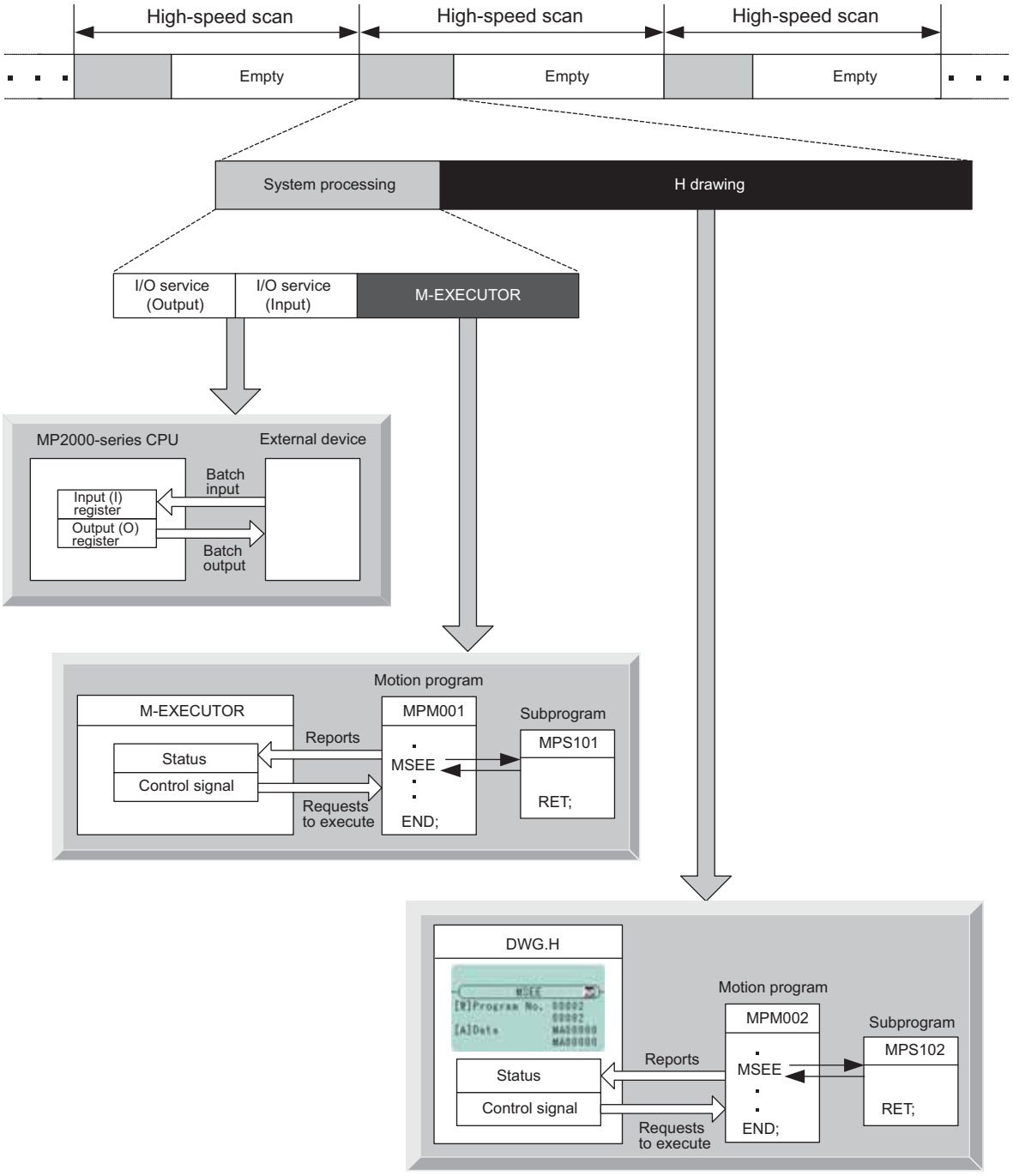
Register motion programs in the M-EXECUTOR program execution definition. A control register (I/O register) is used to start or stop the registered motion program.



INFO M-EXECUTOR is a software module to execute motion programs and sequence programs.

1.5 Motion Program Execution Timing

Motion programs are executed in full synchronization with MP2000 high-speed scans. In every high-speed scan cycle, I/O services are performed first, and the motion program registered in M-EXECUTOR is executed.
Next, the motion program initiated in the MSEE command coded in the DWG.H is executed at the timing of the MSEE command execution.
The following diagram illustrates motion program execution timing.

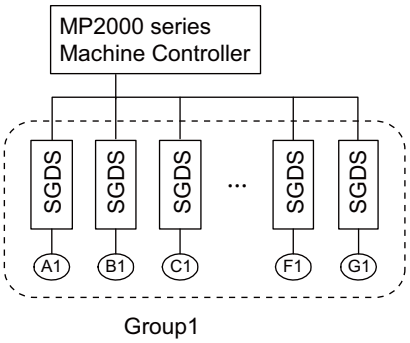


1.6 Grouping

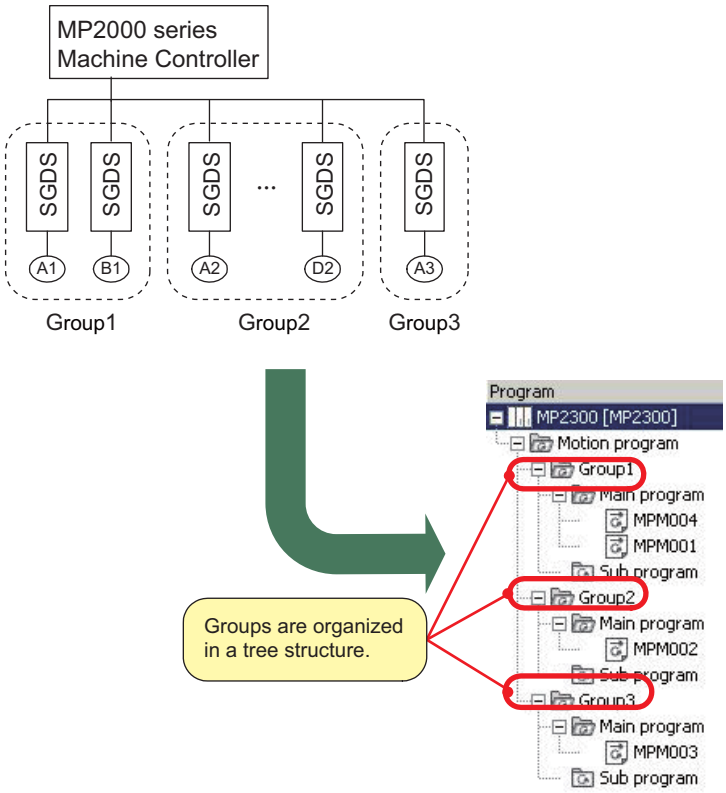
The axes involved in related operations are organized into individual groups. Motion programs can be created for each group. This allows one MP2000-series Machine Controller to independently control multiple machines using group operation. Group operation can be single group operation or multiple group operation.

Definitions for axes to be grouped together are made under *Group Definitions*.

■ Single Group Operation



■ Multiple Group Operation



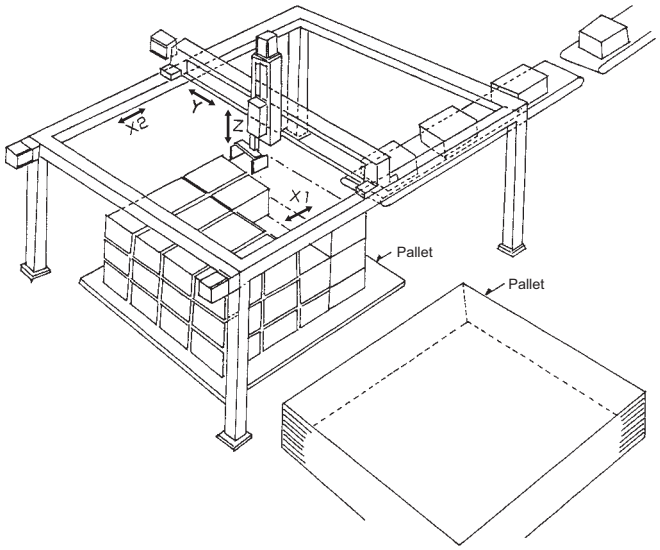
1.7 Application Examples

Motion programs can be used for operations of various systems.
Some application examples are shown below.

1.7.1 Example 1: Handling System

Outline

- To stack a specified number of cardboard boxes on a pallet and transport them to the next process
- The system operation includes three axes motion control for the palletizing process and an automatic pallet feeding sequence.



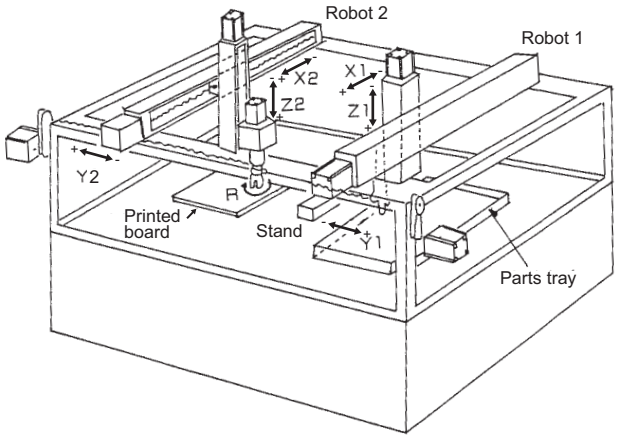
Control points

- Moves X1 and X2 axes in synchronization using a virtual axis.
- Realizes smooth movements by using interpolation.
- Palletizes by calculating the position data with the motion program according to predefined conditions (box dimensions, the number of boxes in a horizontal row, the number of boxes in a vertical row, and the number of boxes in a stack.

1.7.2 Example 2: Mechanical Parts Inserting Machine

Outline

- To insert parts, such as connectors, in a printed board.
- The handling robot takes out the parts and brings them to the stand. The inserting robot inserts the parts in the specified position and angle on the board.



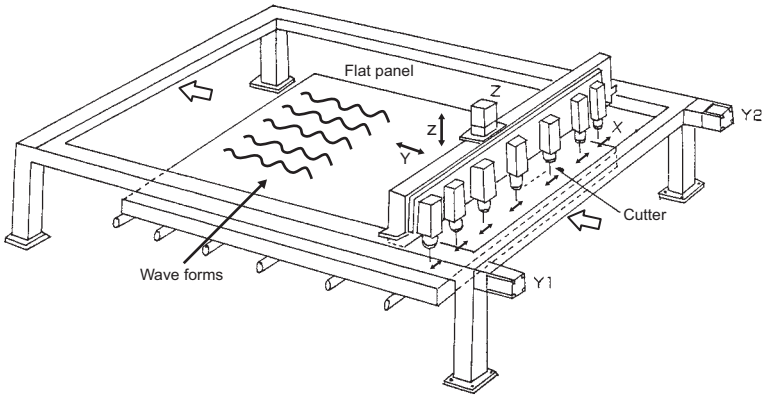
Control points

- Two groups of axes are organized, and programs are created for each group, so that each robot is independently controlled.
- The tact time can be shortened by using two-axes or three-axes linear interpolation.

1.7.3 Example 3: Panel Processing Machine

Outline

- To draw waveforms on a flat panel made of construction material.
- More than ten cutters are mounted in series on the X axis, and the width of the pattern can be easily changed.



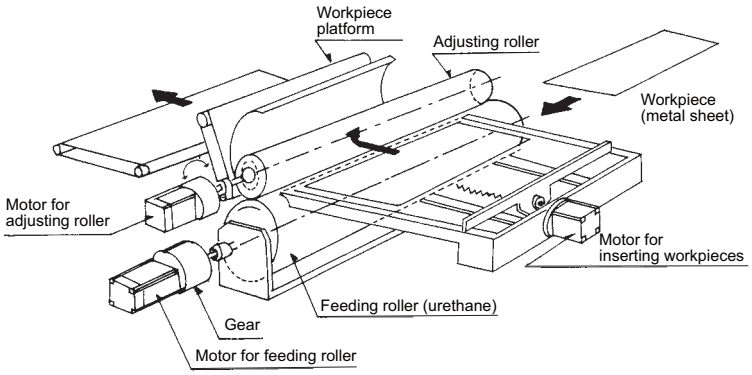
Control points

- Moves X and Y axes in circular interpolation to draw waveforms.
- Moves Y1 and Y2 axes in synchronization using a vertical axis.

1.7.4 Example 4: Metal Sheet Bending Equipment

Outline

- To bend a metal sheet
- A metal sheet can be bent into various shapes by changing the adjusting axis while feeding a sheet using the rolling axis.



Control points

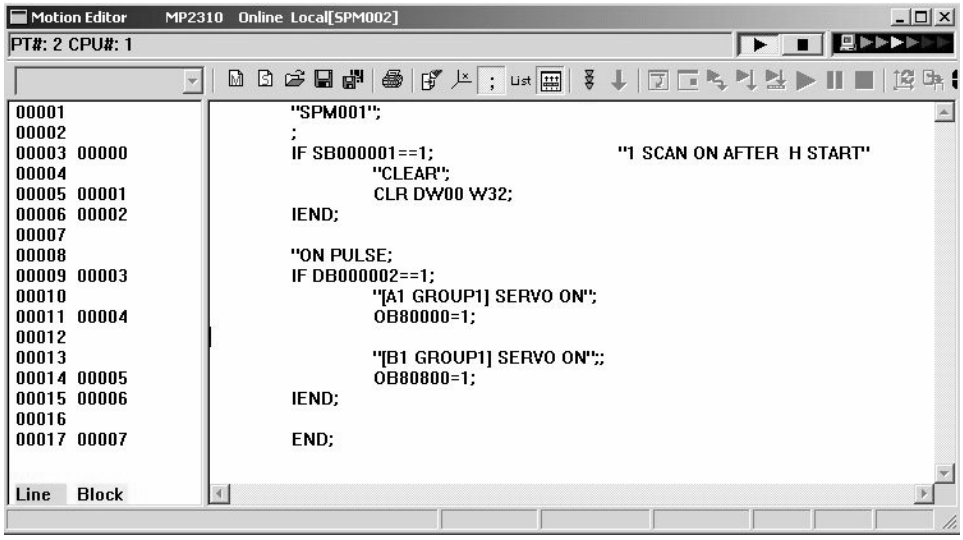
- Controls two axes, a linear axis and rotational axis, in linear interpolation.
- Switches the motion program to be called according to the process.

1.8 What is a Sequence Program?

The sequence program is a scan execution type program written in the language commonly used for the motion program.
An application to cyclically check a status, such as an interlock, can be created by using a sequence program.
A sequence program can be executed by calling from the program execution registry screen dialog box of M-EXECUTOR Module.

Note: M-EXECUTOR Module compatible Machine Controllers are limited to the MP2100, MP2100M, MP2300S, MP2310, and MP2400. The M-EXECUTOR Module cannot be used with other Machine Controllers.

A total of up to 256 sequence and motion programs can be created.
An example of a sequence program is shown below.

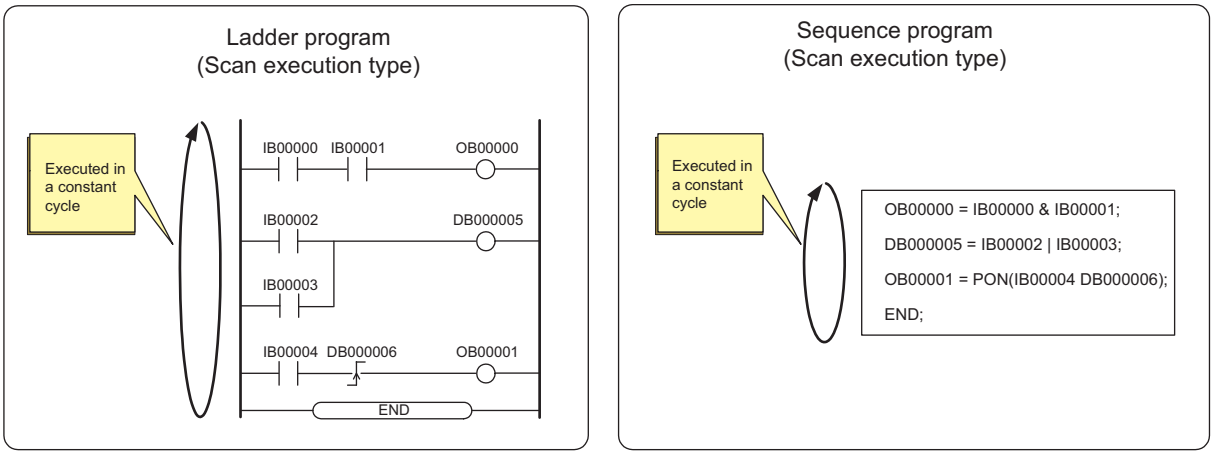


The features of sequence programs are described, starting from the next page.

1.9 Sequence Program Features

1.9.1 Execution Method

A sequence program employs the same execution method as the ladder program.
A sequence program is a cyclically executed scan execution type program. Processing from the program start to an END command is completed within one scan.
Sequence programs can be used by registering them in the program execution registry screen dialog box of M-EXECUTOR Module.

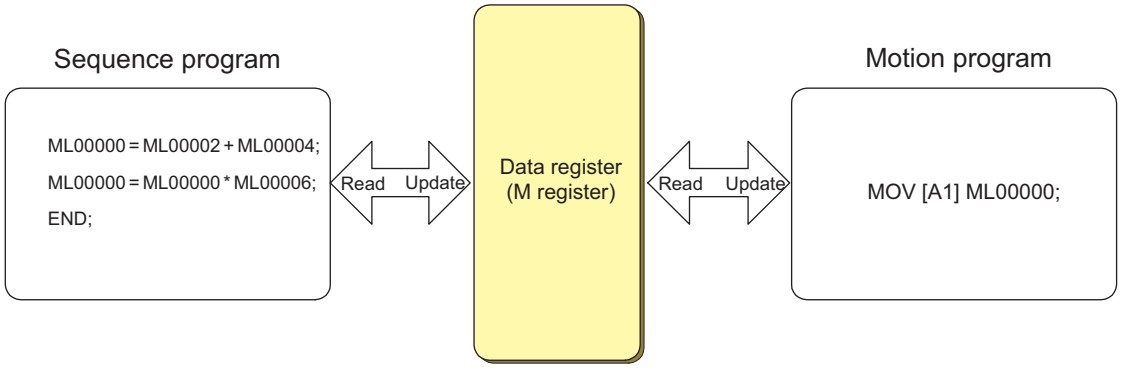


1.9.2 Programming Language Commonly Used in Motion Programs

A sequence program employs the same motion language as a motion program.
The motion language commands that can be used in sequence programs, however, are limited to sequence commands, such as math commands. Commands for motion control, such as axis move commands, cannot be used.
The use of sequence programs allows you to create an application for sequence control without using a ladder program.

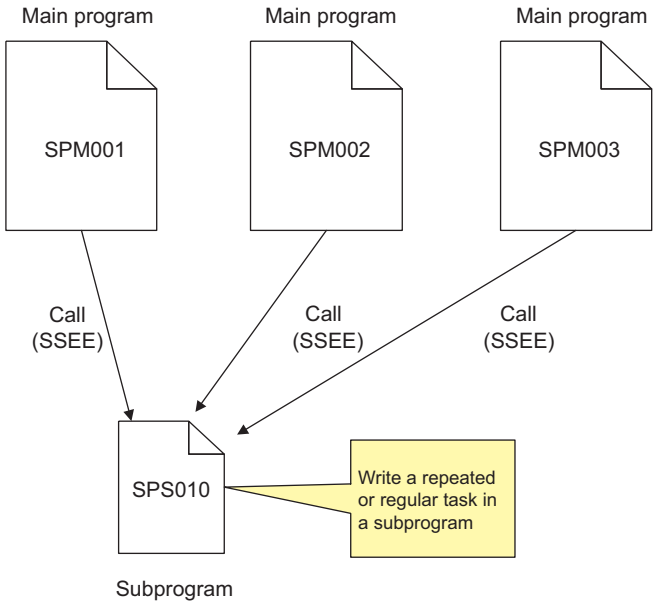
1.9.3 Data Transfer from/to Motion Program

Data can be transferred between a sequence program and a motion program.
Data registers (M registers) are used to transfer the data.
In this way, data updated in the sequence program can be used in the motion program, and vice versa.



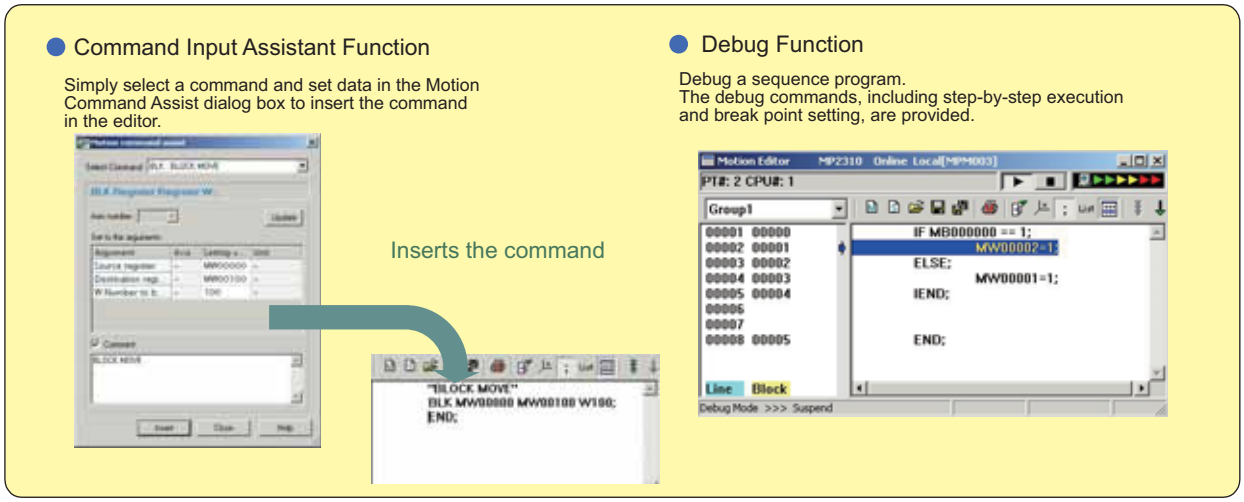
1.9.4 Memory Usage Reduced by Use of Subprograms

Subroutines (subprograms) can be created within a sequence program.
The number of program steps can be minimized by creating a subprogram that includes a set of commands to perform a repeated or regular task, thus reducing memory usage.



1.9.5 Easy Programming Functions (MPE720 Ver.6.04 or later)

The following easy programming functions can also be used for sequence programs.



2

Specifications

This chapter describes the relevant specifications of motion program and engineering tool MPE720.

2.1 MP2000 Series Machine Controller Specifications ----------2-2
2.1.1 Applicable Machine Controller Models ---------- 2-2
2.1.2 Applicable Motion Modules ---------- 2-2
2.1.3 List of Machine Controller Specifications ---------- 2-3
2.2 Engineering Tool MPE720 Specifications ----------2-5
2.2.1 Applicable Version Numbers of the Engineering Tool MPE720 ---------- 2-5
2.2.2 List of Engineering Tool MPE720 Specifications ---------- 2-5
2.3 List of Motion Language Commands ----------2-6

2.1 MP2000 Series Machine Controller Specifications

2.1.1 Applicable Machine Controller Models

Motion programs can be used with the following MP2000-series Machine Controller models.

- MP2100
- MP2100M
- MP2200/CPU-01
- MP2200/CPU-02
- MP2300
- MP2300S
- MP2310
- MP2400
- MP2500
- MP2500D
- MP2500M
- MP2500MD



With MP2100, MP2100M, MP2300S, MP2310, and MP2400 Machine Controllers, both motion programs and sequence programs can be used.

If using the M-EXECUTOR module or sequence programs with MP2100 or MP2100M, the programming tools with following versions are required.

MP2000 Series Controller	Applicable Version
MP2100 MP2100M	Ver 2.66 or later

MPE720	Applicable Version
MPE720 Ver 5	MPE720 Ver 5.44 or later
MPE720 Ver 6	MPE720 Ver 6.10 or later MPE720 Ver 6.10 Lite or later

2.1.2 Applicable Motion Modules

The following motion modules support motion programs.

The axes connected to the following motion modules can be controlled using motion programs.

- Built-in SVB (Built in MP2100, MP2100M, MP2300, MP2300S, MP2310, MP2400, and MP2500, MP2500D, MP2500M, MP2500MD as a standard feature)
- SVR (Mounted on all models of MP2000-series Machine Controllers as standard)
- SVA-01
- SVB-01
- PO-01

2.1.3 List of Machine Controller Specifications

		MP2100, MP2100M	MP2300	MP2300S	MP2400	MP2200 /CPU-01	MP2310	MP2200 /CPU-02	Remarks
Program Capacity		5.5 MB				7.5 MB		11.5 MB	Total user program capacity including ladder programs, motion programs, and sequence programs.
Ladder Program		Applicable			N/A	Applicable			–
	Start Processing	64 drawings max.							–
	Interrupt Processing	64 drawings max.							–
	High-speed Processing	200 drawings max.							–
	Low-speed Processing	500 drawings max.							–
	User function	500 drawings max.							–
Motion Program		Applicable							–
	Number of Programs	256 programs max.							Up to a total of 256 motion programs and sequence program.
	Number of Groups	Eight groups							Up to 16 axes can be set for one group.
	Number of Tasks	16 tasks							Number of motion programs that can be executed simultaneously.
	Number of Parallel Processes (Per Task)	Eight parallel processes							Parallel execution of four main programs × parallel execution of two subprograms.
	Execution Method	• By writing an MSEE instruction in the ladder program • By using an M-EXECUTOR Module (Only for MP2100, MP2100M, MP2300S, MP2310, and MP2400)							–
	Starting Method	The program starts running at the rising edge of control signal bit 0 (Program start request).							–
	Override	Can be set in the range from 0.01% to 327.67%.							–
	Operation Mode	ABS (absolute) and INC (incremental) mode							Mode switching by use of the exclusive command (ABS/INC).
	Reference Unit	• Built-in SVB, SVB-01, and SVR Module: pulse, mm, deg, inch, µm • SVA-01 and PO-01 Module: pulse, mm, deg, inch							–
	Min. Reference Unit	• E pulse 1 • mm, deg, inch, µm 1, 0.1, 0.01, 0.001, 0.0001, 0.00001							–
	Reference Range	-2147483648 to +2147483647 (32-bit with sign)							–
	Number of Simultaneously Controlled Axes (Per Task)	16 axes max.							–

		MP2100, MP2100M	MP2300	MP2300S	MP2400	MP2200 /CPU-01	MP2310	MP2200 /CPU-02	Remarks
		Applicable	N/A	Applicable		N/A	Applica-ble	N/A	–
Sequence Program	Number of Programs	256 programs max. (The execution timing can be selected from the start drawing, high-speed scan drawing, or low-speed scan drawing.)							Up to a total of 256 motion programs and sequence.
	Number of Tasks	16 tasks max.							Number of sequence programs that can be executed simultaneously.
	Number of Parallel Processings (Per Task)	None							–
	Execution Method	By using the M-EXECUTOR Module							–
	Start Method	Started by the system.							Started by the system by registering programs in the M-EXECUTOR module.
Accessible Registers	M Register	Accessible (65535 words)							Memory backed up by battery.
	S Register	Accessible (8192 word)							Memory backed up by battery.
	I Register	Accessible (32768 words and motion monitoring parameters)							–
	O Register	Accessible (32768 words and motion setting parameters)							–
	C Register	Accessible (16384 words)							–
	D Register	Accessible (Can be specified in the range from 0 to 16384 words)							Internal register unique to each drawing. Can be used by only the corresponding drawing.
	# Register	Accessible only from ladder program (Can be specified in the range from 0 to 16384 words) (# register cannot be accessed from motion program and sequence program.)							–

2.2 Engineering Tool MPE720 Specifications

2.2.1 Applicable Version Numbers of the Engineering Tool MPE720

Motions programs and/or sequence programs can be created or edited on the following engineering tool MPE720 versions.

- MPE720 Ver.5 (Compatible with MP2000-series Machine Controller models excluding MP2400)
- MPE720 Ver.6 (Compatible with all models of MP2000-series Machine Controllers)
- MPE720 Ver.6 Lite (Compatible only with MP2400)



The above engineering tool software can be installed on one personal computer.

2.2.2 List of Engineering Tool MPE720 Specifications

		MPE720 Ver.5 (CPMC-MPE720)	MPE720 Ver.6 (CPMC-MPE770)	MPE720 Ver.6 Lite (CPMC-MPE770L)	Remarks
Machine Controller Model	MP2100	Compatible		Not compatible	–
	MP2100M	Compatible		Not compatible	–
	MP2200/CPU-01	Compatible		Not compatible	–
	MP2200/CPU-02	Compatible		Not compatible	–
	MP2300	Compatible		Not compatible	–
	MP2300S	Compatible		Not compatible	–
	MP2310	Compatible		Not compatible	–
	MP2400	Not compatible	Compatible		–
Programs	Ladder Program	Supported		Not supported	–
	Motion Program	Supported			–
	Sequence Program	Supported			–
Command Input Assistant Function		Not provided	Provided (Ver.6.04 or later)		–
Program Execution Registration Function		Provided (Ver.5.38 or later)	Provided (Ver. 6.04 or later)		Supported only by MP2100, MP2100M, MP2300S, MP2310, and MP2400
Debug Function		Provided			–
Motion Task Manager		Provided			–
Drive Control Panel		Provided (Ver.5.38 or later)	Provided (Ver.6.04 or later)		Supported only by MP2100, MP2100M, MP2300S, MP2310, and MP2400
Test Run Function		Not provided	Provided (Ver.6.04 or later)		–
Axis Operation Monitor/ Alarm Monitor Function		Not provided	Provided (Ver.6.04 or later)		–

2.3 List of Motion Language Commands

Type	Command	Name	Type	Commands	Name
Axis Set Commands	ABS	Absolute Mode	Arithmetic Operation	=	Substitute
	INC	Incremental Mode		+	Add
	ACC	Acceleration Time Change		-	Subtract
	DCC	Deceleration Time Change		*	Multiply
	SCC	S-curve Time Constant Change		/	Divide
	VEL	Set Speed		MOD	Remainder
	FMX	Maximum Interpolation Feed Speed	Logical Operation	|	OR (Logical OR)
	IFP	Interpolation Feed Speed Ratio Setting		&	AND (Logical AND)
	IAC	Interpolation Acceleration Time Change		^	XOR (Logical exclusive OR)
	IDC	Interpolation Deceleration Time Change		!	NOT (Inversion)
Axis Move Commands	MOV	Positioning	Data Comparison	==	Match
	MVS	Linear Interpolation		<>	Mismatch
	MCW	Clockwise Circular/Helical Interpolation		>	Greater than
	MCC	Counterclockwise Circular/Helical Interpolation		<	Less than
	ZRN	Zero Point Return		>=	Greater than or equal to
	SKP	Linear Interpolation with SKIP Function		<=	Less than or equal to
	MVT	Set Time Positioning	Data Operation	SFR	Right Shift
	EXM	External Positioning		SFL	Left Shift
Control Commands	POS	Current Position Set		BLK	Block Move
	MVM	Move On Machine Coordinates		CLR	Clear
	PLN	Coordinate Plane Setting		ASCII	ASCII Conversion 1
	PLD	Program Current Position Update	Basic Functions	SIN	Sine
	PFN	In-Position Check		COS	Cosine
	INP	Set In-Position Check Width		TAN	Tangent
Program Control Commands	IF ELSE IEND	Branch		ASN	Arc Sine
	WHILE WEND	Repeat		ACS	Arc Cosine
	PFORK, JOINTO, PJOINT	Parallel Execution		ATN	Arc Tangent
				SQT	Square Root
				BIN	BCD to Binary
	SFORK, JOINTO, SJOINT	Selective Execution		BCD	Binary to BCD
				S{ }	Set Bit
				R{ }	Reset Bit
	MSEE	Subroutine (motion subprogram) Call		PON	Rising Pulse
	SSEE	Subroutine (sequence subprogram) Call		NON	Falling Pulse
	UFC	User Function Call from Motion Program		TON	On-Delay Timer
				TOF	Off-Delay Timer
	FUNC	User Function Call from Sequence Program	C Language Control Commands	CTSK	C Language Task Control
	END	Program End			
	RET	Subprogram End		CFUNC	C Language Function Call
	TIM	Dwell Time			
	IOW	I/O Variable Wait			
	EOX	One Scan Wait			
	SNGD/SNGE	Disable/Enable Single Block			

3

Program Development Flow

This chapter describes the procedures from system setup to operation start using the programming tool MPE720 version 6.

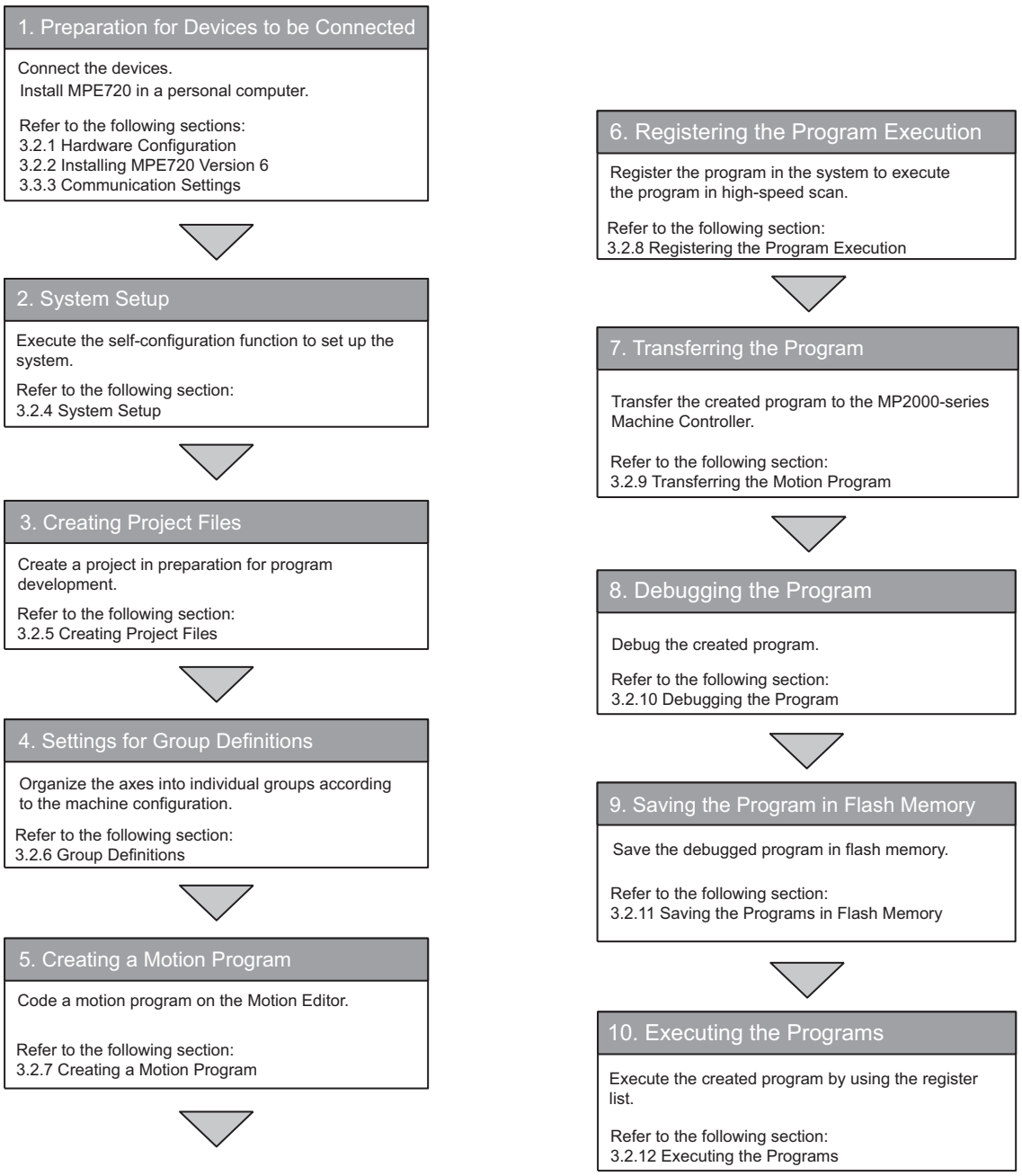
3.1 Program Development Flow ----------3-2

3.2 Program Development Procedure ----------3-3

3.2.1 Hardware Configuration ---------- 3-3
3.2.2 Installing MPE720 Version 6 ---------- 3-3
3.2.3 Communication Settings ---------- 3-3
3.2.4 System Setup ---------- 3-3
3.2.5 Creating Project Files ---------- 3-4
3.2.6 Group Definitions ---------- 3-5
3.2.7 Creating a Motion Program ---------- 3-6
3.2.8 Registering the Program Execution ---------- 3-7
3.2.9 Transferring the Motion Program ---------- 3-10
3.2.10 Debugging the Program ---------- 3-12
3.2.11 Saving the Programs in Flash Memory ---------- 3-13
3.2.12 Executing the Programs ---------- 3-14

3.1 Program Development Flow

In this chapter, motion program development procedures are described according to the following flowchart.



Note: 1. The development procedure for sequence programs is basically the same as that for motion programs. For this reason, descriptions of the sequence program development flow are omitted.
2. The above flowchart shows an example of program development. To use programs for the actual system, settings for the external devices are required.

■Motion Program to be Created

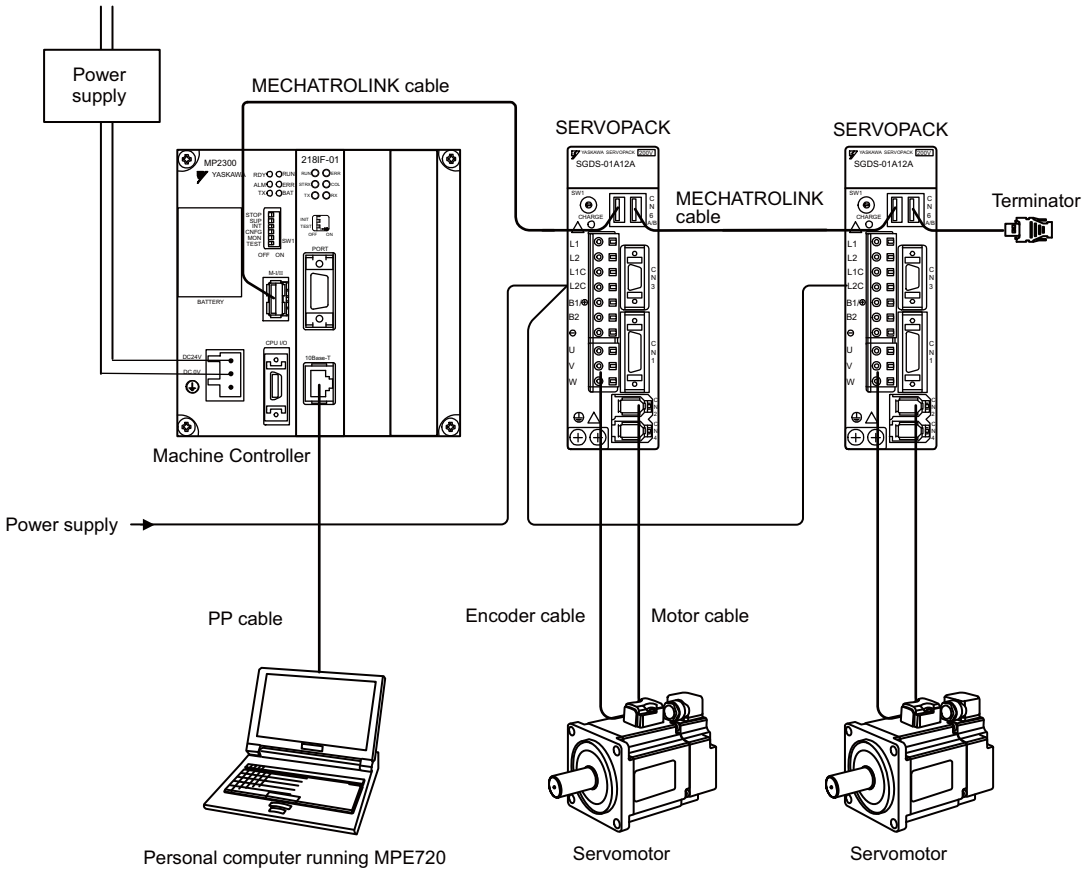
In this chapter, the following motion program is used to explain the program development flow. The motion program contains only three lines for the simple operation of moving an axis 150,000 pulses from the current position and then stopping.

```
INC;                          “INCREMENTAL MODE”
MOV [A1]150000 [B1]150000;    “2 AXES 150000 PULSES POSITIONING”
END;
```

3.2 Program Development Procedure

3.2.1 Hardware Configuration

The program development procedure is explained using the following system configuration.



Note: In the system configuration above, set the station numbers of the two SERVOPACKs to 1 and 2.

3.2.2 Installing MPE720 Version 6

Install MPE720 Ver.6 in a personal computer.

For information on the installation procedure, refer to *Engineering Tool for MP2000 Series Machine Controller MPE720 Version 6 User's Manual* (manual no.: SIEPC88070030).

3.2.3 Communication Settings

Set the conditions for communications between the computer on which MPE720 Ver.6 is installed and the MP2000-series Machine Controller.

For information on the communication settings, refer to *Engineering Tool for MP2000 Series Machine Controller MPE720 Version 6 User's Manual* (manual no.: SIEPC88070030).

3.2.4 System Setup

Use the self-configuration function to setup the system. The self-configuration function automatically recognizes the modules installed on the MP2000-series Machine Controller and the devices connected to the Machine Controller through the MECHATROLINK connection. This function allows you to quickly and easily set up the system. The self-configuration can be executed either when the power to the Machine Controller turns ON or by using MPE720.

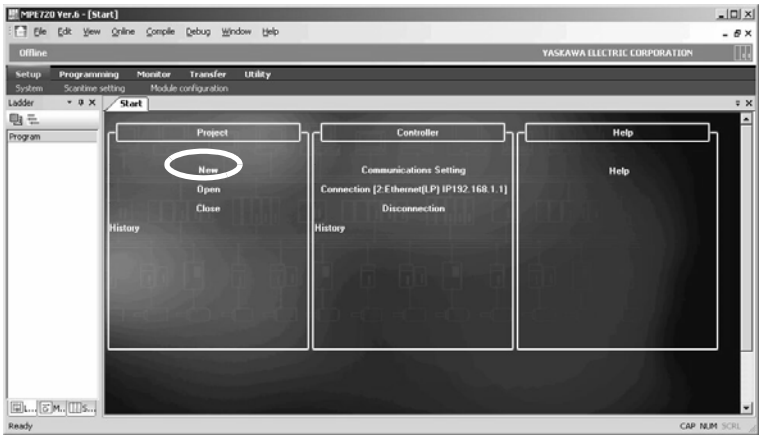
For information on how to execute self-configuration, refer to the user's manual for the Machine Controller to be used.

3.2.5 Creating Project Files

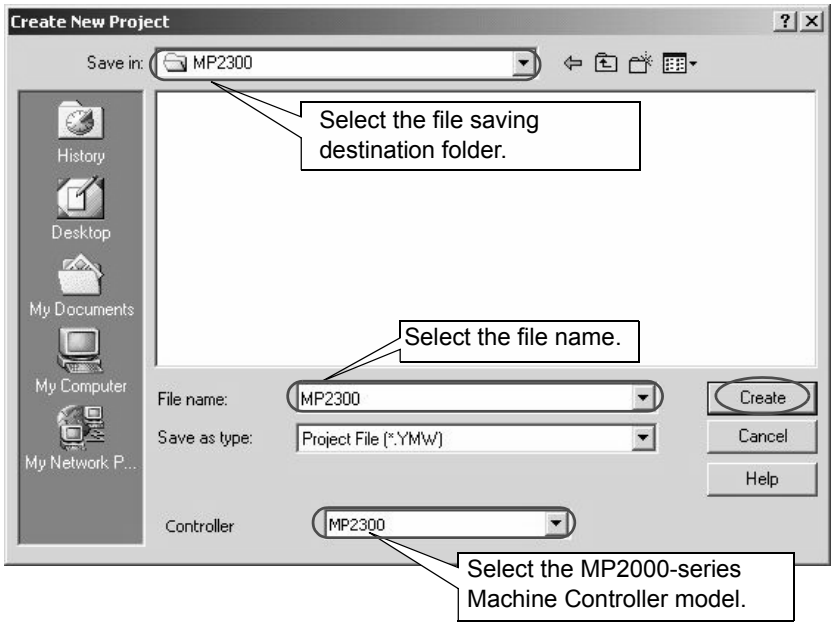
1. Double-click the **MPE720 Ver.6** icon on the computer desktop to launch the MPE720 Ver. 6.



2. Click ***New*** under ***Project***.



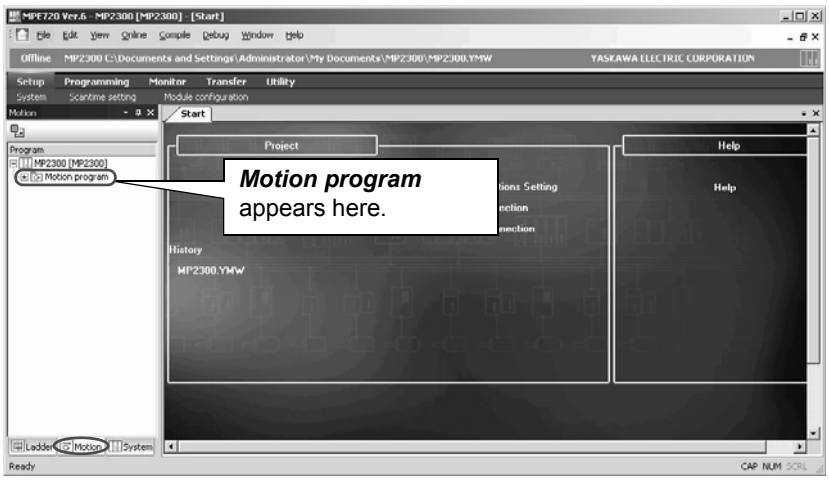
3. Specify the file name, file saving destination folder, and Machine Controller model. Then click the **Create** button.



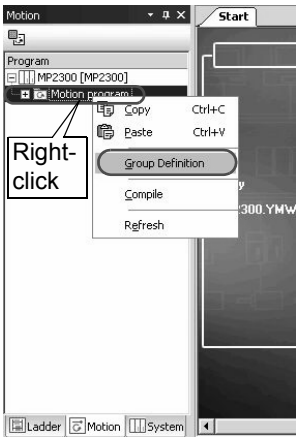
3.2.6 Group Definitions

Before creating a motion program, organize the axes into individual groups according to the machine configuration.

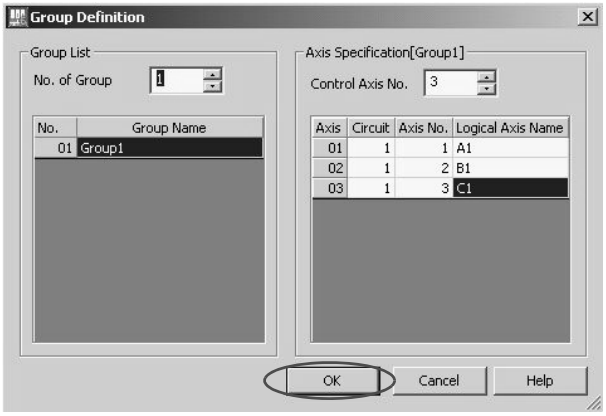
1. Click the **Motion** tab to display **Motion program** in the subwindow.



2. Right-click **Motion program** in the subwindow. Select **Group Definition** from the drop-down menu.



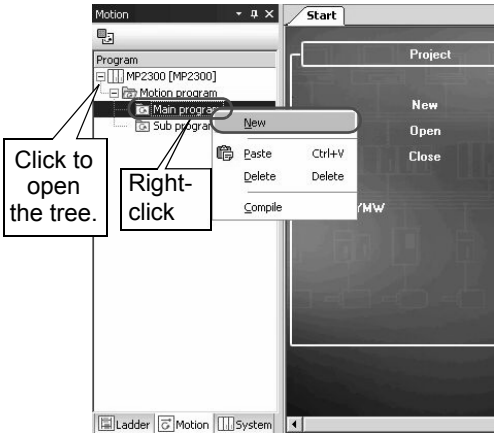
3. Click **OK**. For details on group definitions, refer to *7.3 Group Definition*.



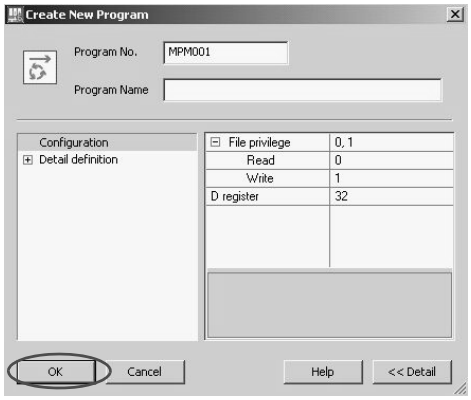
3.2.7 Creating a Motion Program

Start the Motion Editor to create a motion program.

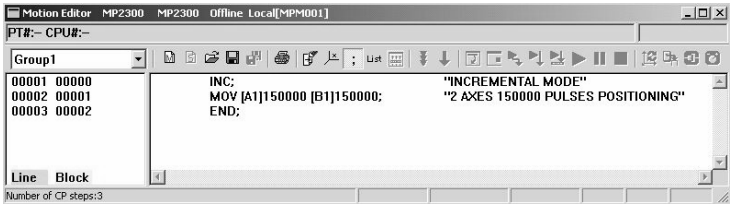
1. Programs are displayed under **Motion program** in the subwindow. Right-click **Main program**. Select **New** from the drop-down menu.



2. Click **OK**.



3. Enter the sample motion program provided in *3.1 Program Development Flow*.



4. Click the **Save** icon ( ) on the toolbar of the **Motion Editor** window to start compiling. The motion program will be automatically saved after compiling is completed.

IMPORTANT

Note that the motion program will not be automatically saved if the Error List dialog box appears during compiling.

3.2.8 Registering the Program Execution

Call the created motion program from the H drawing using an MSEE command. For details, refer to *4.3.2 Registering the Program Execution*.

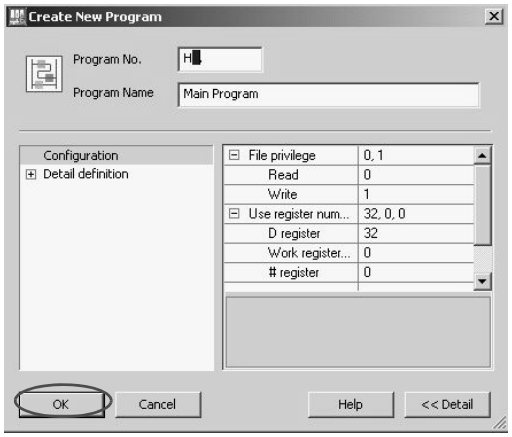
1. Click the Ladder tab in the subwindow to display **Ladder program**.



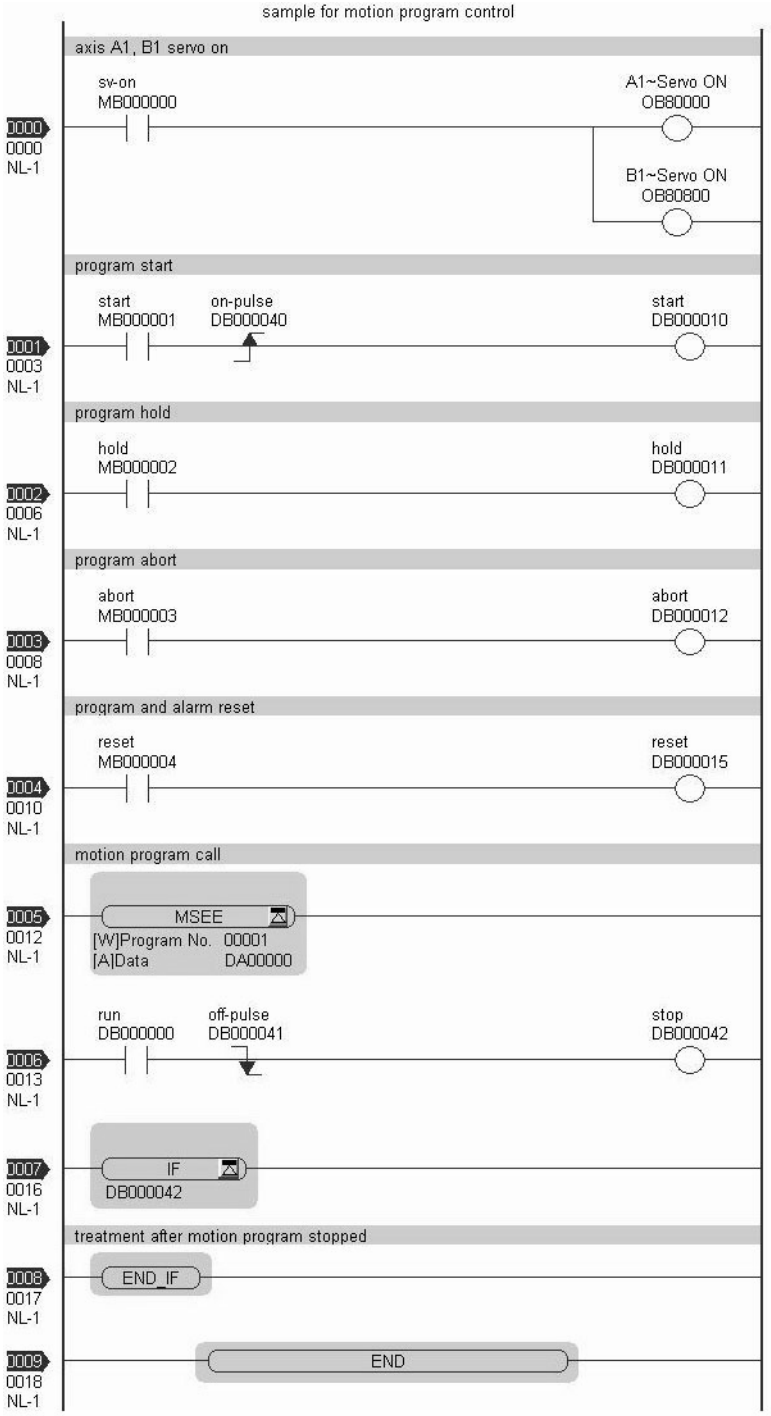
2. Right-click **High-speed** under **Ladder program** in the subwindow. Select **New** from the drop-down menu.



3. Click **OK**.



4. Create the ladder program shown below. After the ladder program has been created, compile the program by pressing the **F8** key on the keyboard or clicking the  icon on the toolbar.



INFO

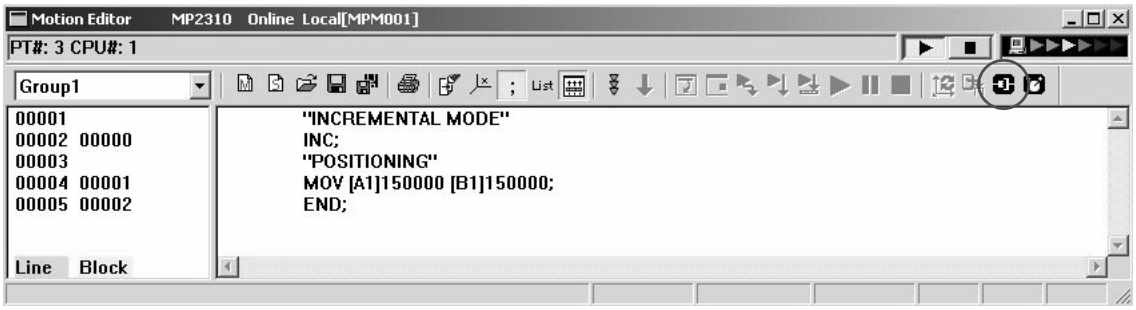
- Make sure that bit 0 of motion monitoring parameter IW□□00 (Operation Ready) is ON before turning ON the Servo ON command MB000000.
- The Servo ON command will not be accepted if the Operation Ready bit is OFF.

INFO

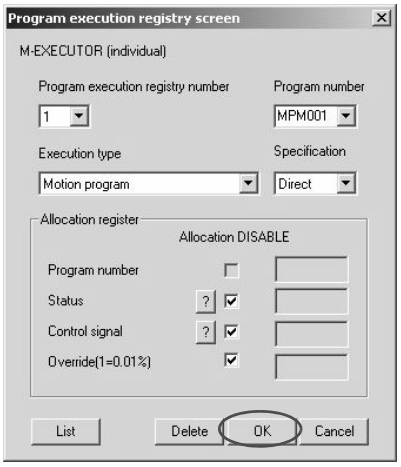
The motion programs can be registered to run by registering the programs in the M-EXECUTOR program execution definition, without creating the ladder program described on the previous page.

The procedure to register motion programs to the program definition of M-EXECUTOR is described below. Before using this procedure, be sure to carry out the operation described in *3.2.9 Transferring the Motion Program*.

1. Click the  icon on the toolbar in the **Motion Editor** window where the motion program is created.



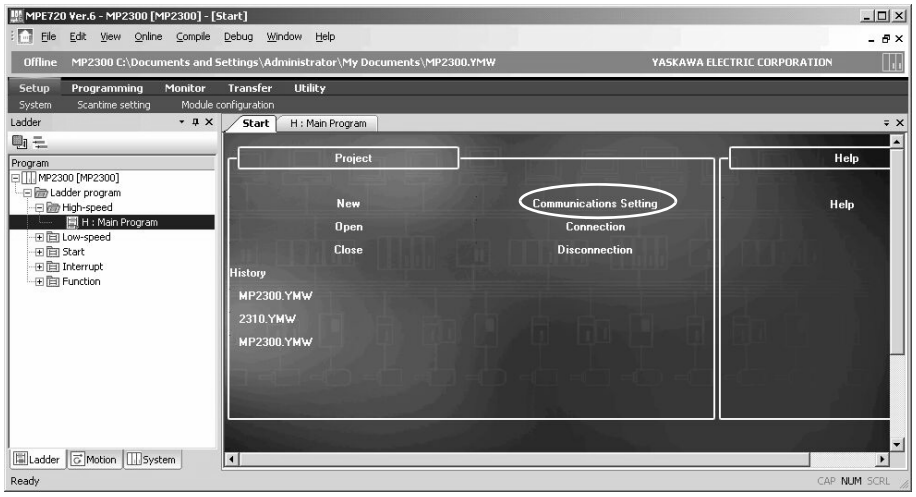
2. The **Program execution registry screen** dialog box will open. Click **OK** to register the program.



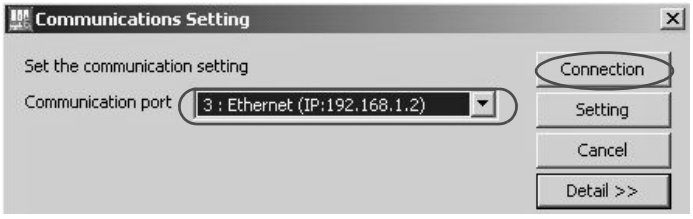
3.2.9 Transferring the Motion Program

Transfer the motion program to the MP2000-series Machine Controller. If the program is created on a computer using MPE720 Version 6 and the Machine Controller connected online, this operation will not be required.

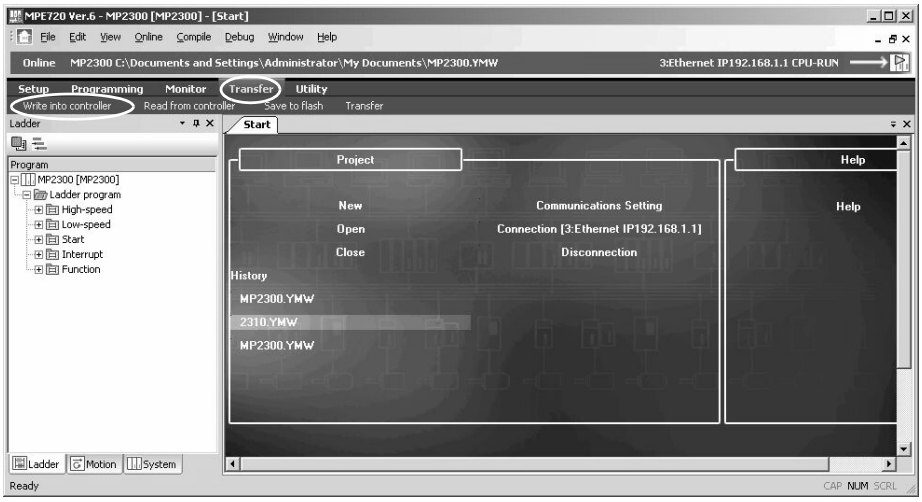
1. Click **Communications Setting** in the following window.



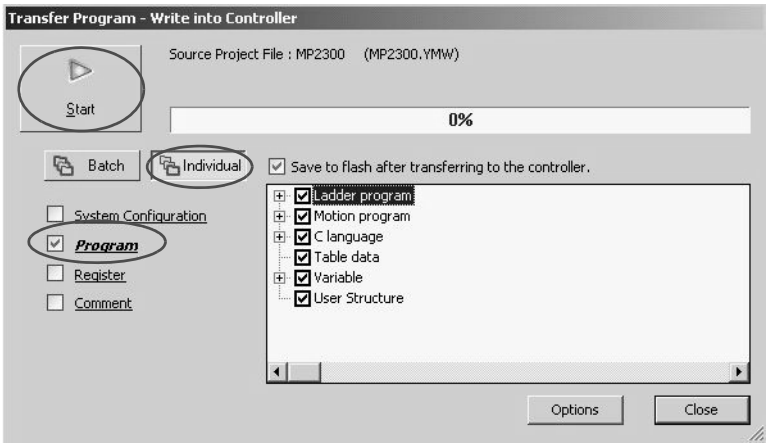
2. Select the communication port selected in *3.2.3 Communication Settings*, and click the **Connection** button.



3. **Offline** will change to **Online**. Select ***Transfer - Write into controller***.



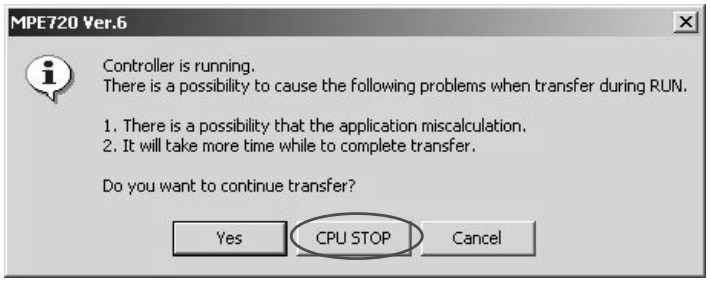
4. Click the **Individual** button, and then select the ***Program*** check box. Click the **Start** button.



INFO

- When **Individual** transfer is selected, the same file in the Machine Controller will be overwritten with the selected project file data.
- When **Batch** transfer is selected, the MP2000-series Machine Controller's RAM will be cleared before transfer, and all project file data will be written in the RAM.

5. Click the **CPU STOP** button to start transfer.



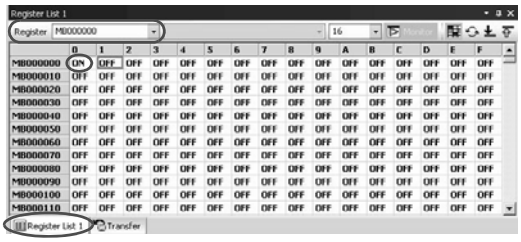
6. Click the **Yes** button in the following dialog box to restart the Machine Controller.



3.2.10 Debugging the Program

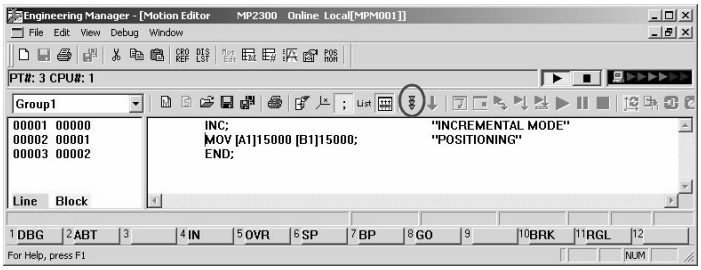
Debug the created program. For details on debugging, refer to *9.4 Debug Function*.

1. Click the **Register List 1** tab to display the register list. Specify register MB000000. Set MP000000 to ON as follows to turn the servo ON.

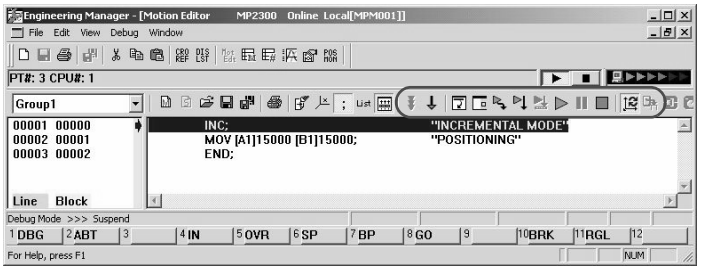


INFO When using the M-EXECUTOR as described in *3.2.8 Registering the Program Execution*, directly set the motion setting parameter to turn the servo ON.

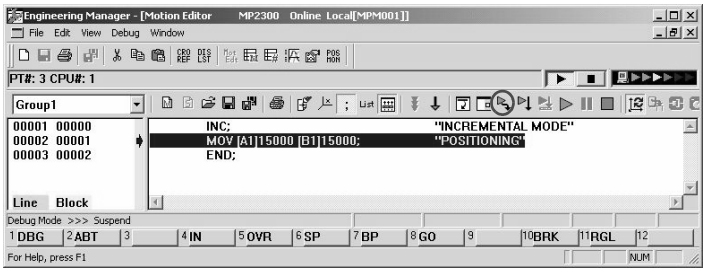
2. Click the **Debug Mode** icon ( ) on the toolbar.



3. The Motion Editor will enter debug mode.



4. Click the **Step In** icon ( ) to execute the program line by line, and check the programmed movements. For details on debugging, refer to *9.4 Debug Function*.

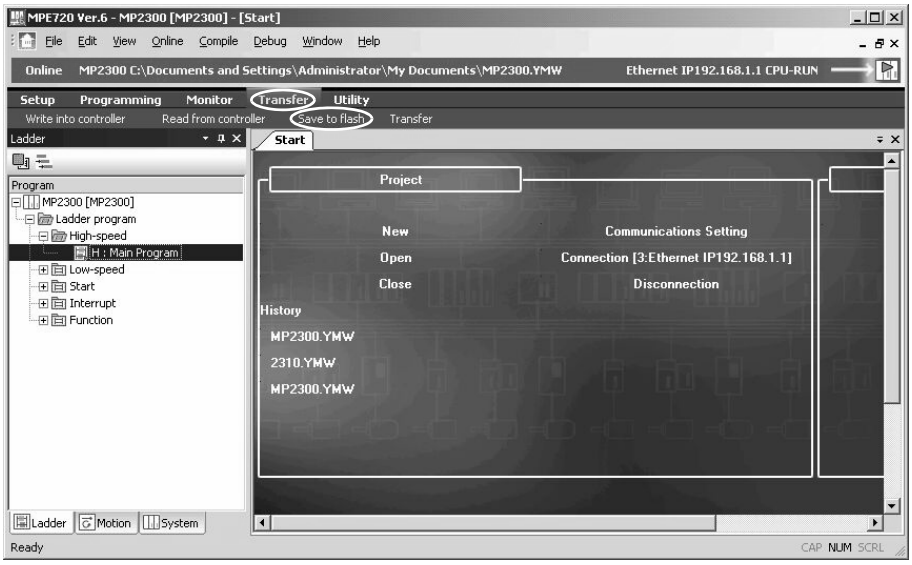


5. Debug the program until the END command. When the debugging operation is completed, turn the servo OFF.

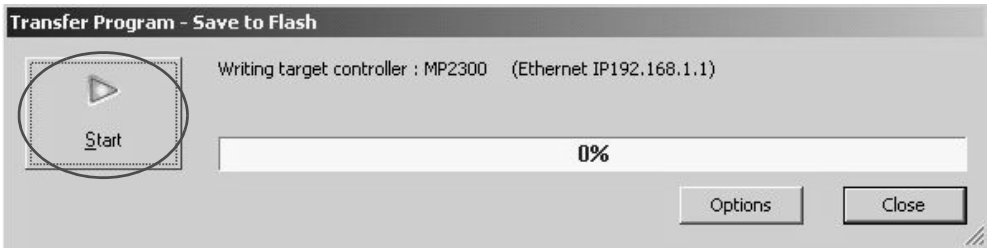
3.2.11 Saving the Programs in Flash Memory

Save the data in the MP2000-series Machine Controller's RAM to the flash memory.

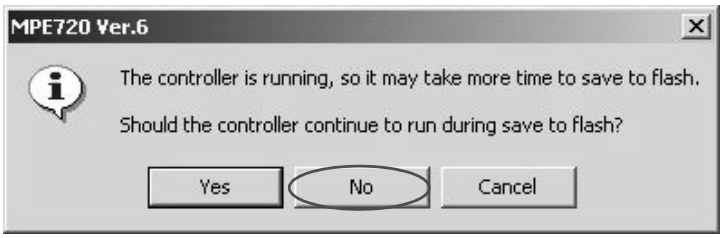
1. Select ***Transfer - Save to flash*** in the following window.



2. Click the **Start** button.



3. Click the **CPU STOP** button to start saving.



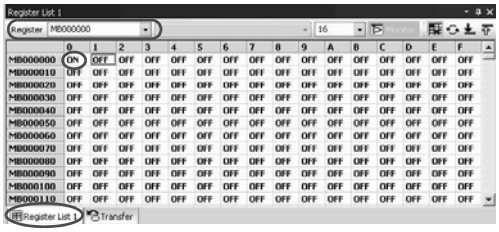
4. Click the **Yes** button in the following dialog box to restart the Machine Controller.



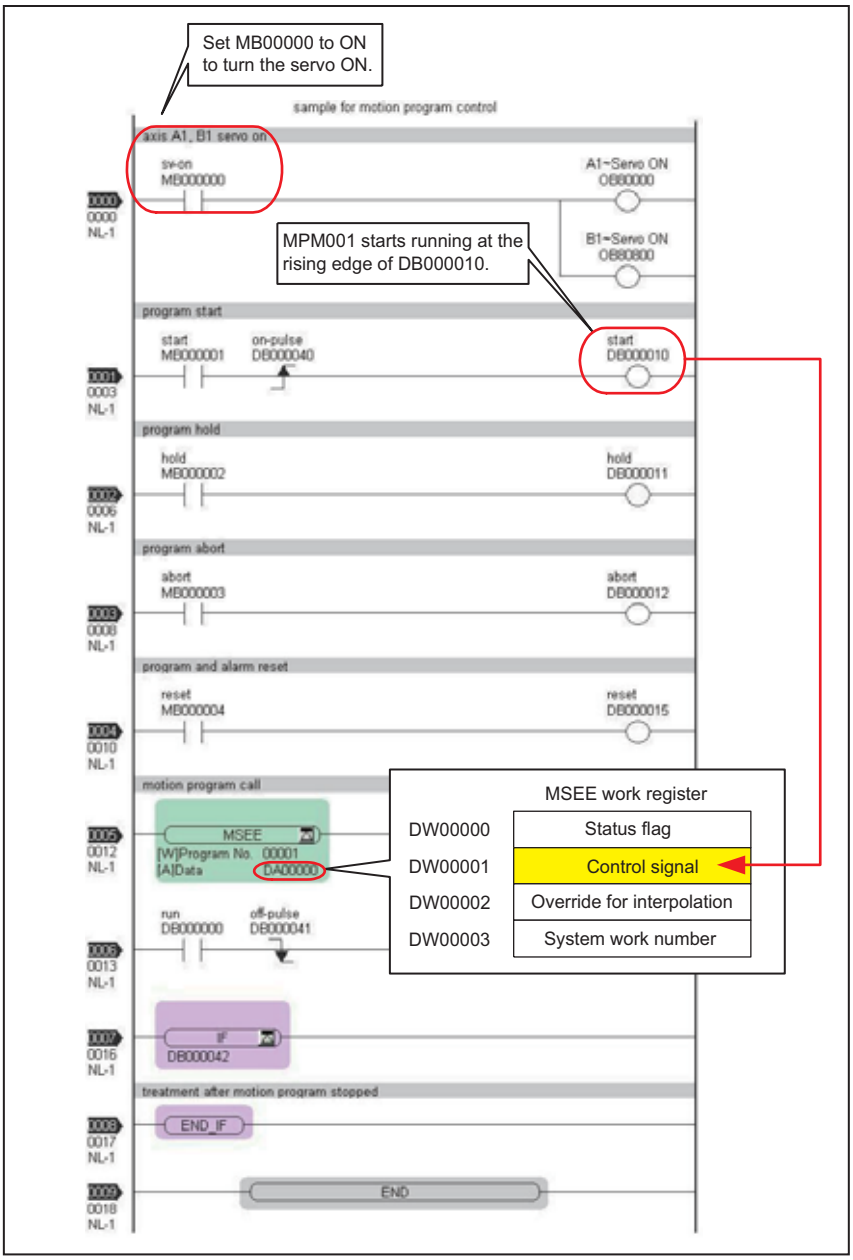
3.2.12 Executing the Programs

Execute the created programs to operate the actual machine. Turn the Program Start Request bit to ON using the control signal to execute the motion program.

1. Click the **Register List 1** tab to display the register list. Specify register MB000000. Set MB000000 to ON to turn the servo ON.



2. Set MB000001 in the register list to ON to execute motion program MPM001.



4

Motion Programs

This chapter describes motion program types and how to run them.

4.1 Types of Motion Programs - - -

4.1 Types of Motion Programs

There are two motion program types, as listed below.

Type	Designation Method	Features	No. of Programs
Main Program	MPM□□□ (□□□= 1 to 256)	• Called from the M-EXECUTOR program execution definition • Called from DWG.H	Up to 256 programs including following programs can be created. • Motion main program • Motion subprogram • Sequence main program • Sequence subprogram
Subprogram	MPS□□□ (□□□= 1 to 256)	• Called from the main program	

INFO

■ The program numbers of motion programs are managed in the same manner as the sequence program numbers. Assign a unique number for each program number.

- Program number of Motion program MPM□□□, MPS□□□
- Program number of Sequence program SPM□□□, SPS□□□

■ The MP2000-series Machine Controllers can execute up to 16 motion programs simultaneously. An alarm (no system work error) will occur if 17 or more programs are executed simultaneously.

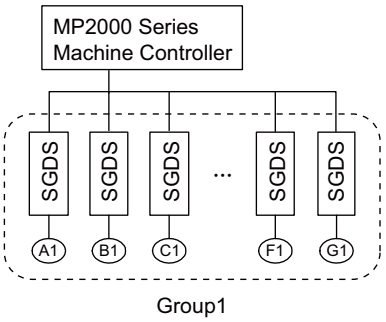
- No system work error: Bit E of the leading word in the MSEE work registers

4.2 Motion Programs For Each Axis Group

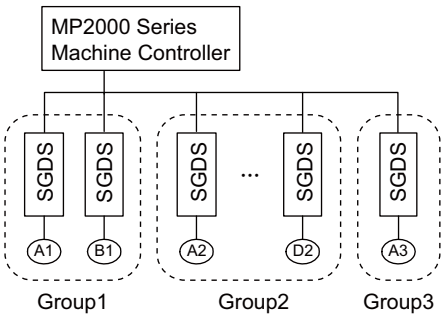
With motion programs, the axes that have related operations are organized into individual groups, and programs can be created for each group. This allows one MP2000-series Machine Controller to independently control multiple machines using group operation. Group operation can be single group operation or multiple group operation. Definitions for axes to be grouped together are made under *Group Definitions*.

For information on the group definition setting window, refer to *7.3 Group Definition*.

■ Single Group Operation



■ Multiple Group Operation



4.3 Running a Motion Program

4.3.1 How to Run a Motion Program

To run the created motion programs, the user must register them in the system. The motion programs registered in the system can be referenced in the high-speed scan cycle.

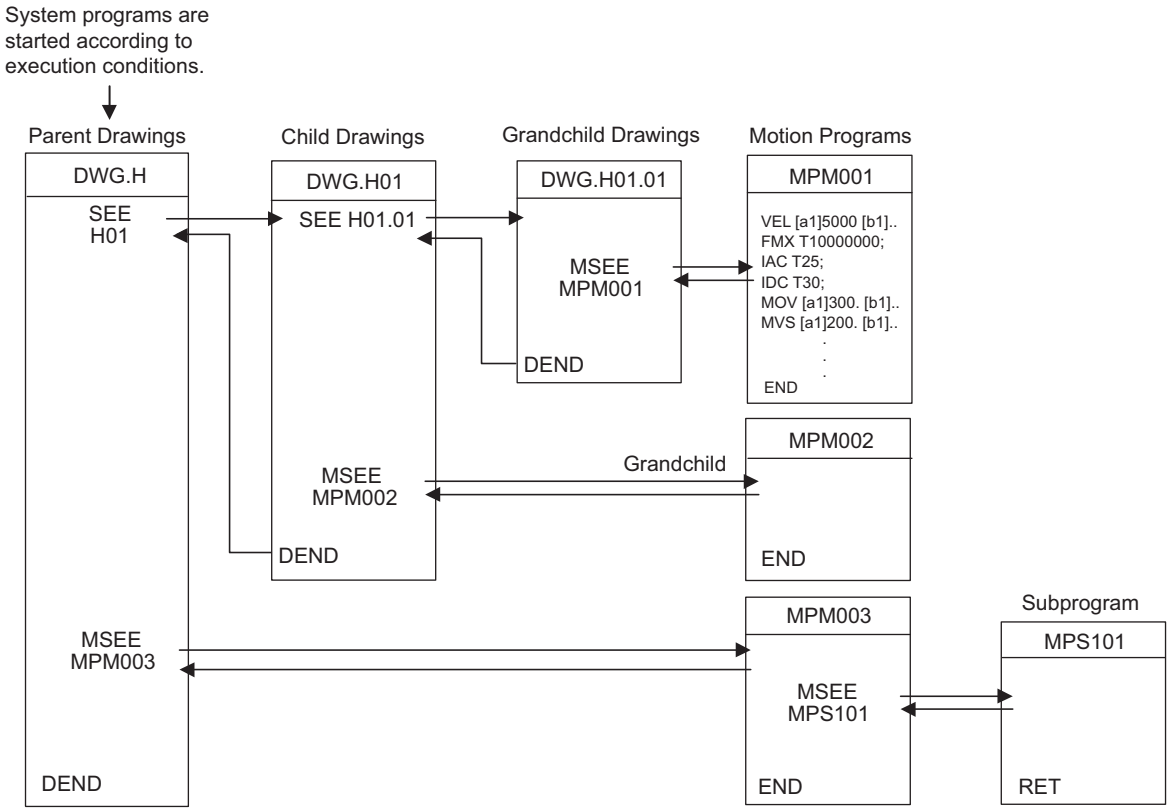
Motion programs can be run in two ways, depending on how they were registered in the system:

- Calling it from the ladder program using a MSEE command
- Registering it to the M-EXECUTOR program execution definition

Now, this section explains each way to run a motion program:

(1) Calling the Motion Program from the Ladder Program Using a MSEE command

After creating a motion program, embed an MSEE command (Motion Program Call command) in the H drawings. Motion programs can be called from any parent, child, or grandchild drawing in an H drawing. The following figure shows an example of motion program execution.



H drawing ladder commands are executed in hierarchical order i.e., parent drawings, child drawings, then grandchild drawings in each high-speed scan cycle.

To start up the motion program, after the MSEE command is incorporated, use a control signal to turn ON the request for the program operation startup.

Motion programs are also called in each scan cycle, but unlike ladder programs, all motion programs cannot be executed in one scan. For this reason, motion programs are executed and controlled by special system's motion management function.

IMPORTANT

When running a motion program, pay attention to the followings:.

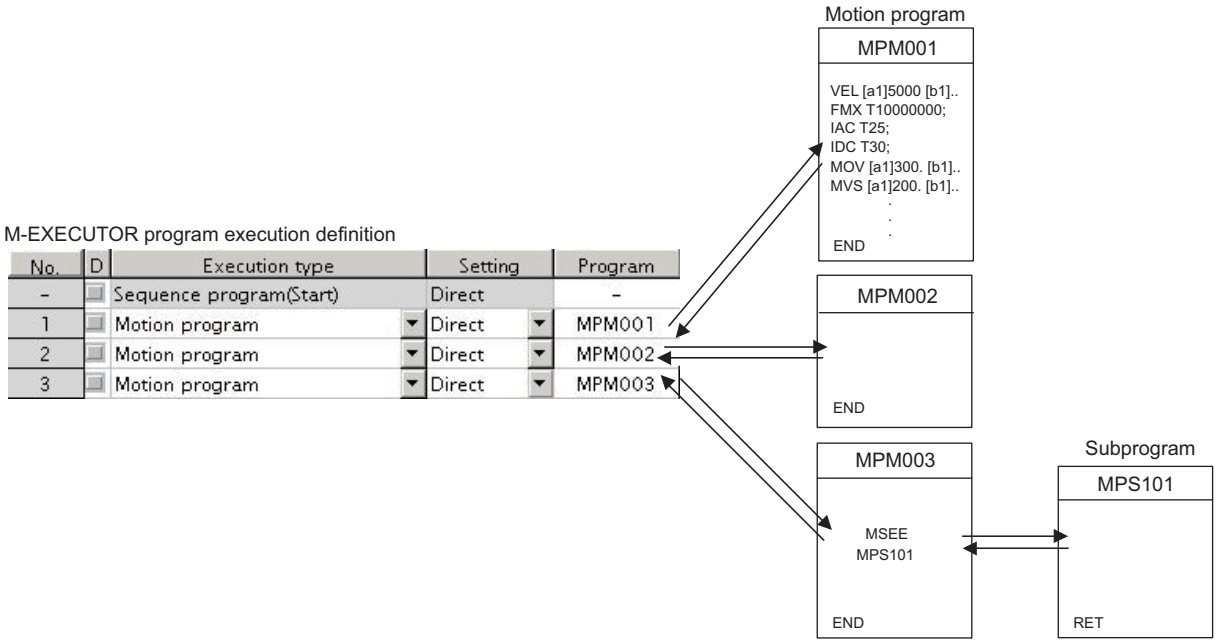
- The motion program registered in M-EXECUTOR cannot be executed using a MSEE command.
- Multiple motion programs with the same number cannot be executed using a MSEE command.
- A subroutine (MPS□□□) cannot be executed from a MSEE command in a ladder.
- It can only be referenced from a motion program (MPM□□□, MPS□□□).
- A sequence program (SPM□□□, SPS□□□) cannot be executed from a MSEE command in a ladder.

(2) Registering it to the M-EXECUTOR Program Execution Definition

After creating a motion program, register it in the M-EXECUTOR program execution definition screen.

The programs registered in the M-EXECUTOR program execution definition screen are executed in ascending numeric order.

The execution example is shown in the figure below.



To start up the motion program, after the motion program registration, use a control signal to turn ON the request for the program operation startup.

The motion program registered in M-EXECUTOR is executed at a scan cycle, but similar to a ladder, the whole program cannot be executed at a single scan. In case of the motion program, a motion management function in the system carries out an execution control exclusive for the motion programs.

IMPORTANT

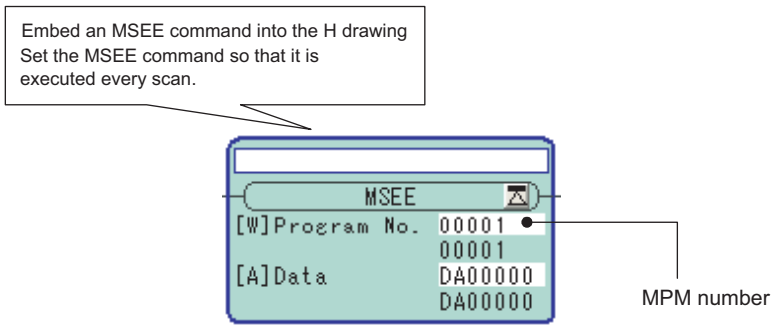
When registering a motion program to M-EXECUTOR, pay attention to the followings:

- Multiple motion programs with the same number cannot be registered.
- Multiple motion programs with the same number cannot be referenced using an indirect designation.

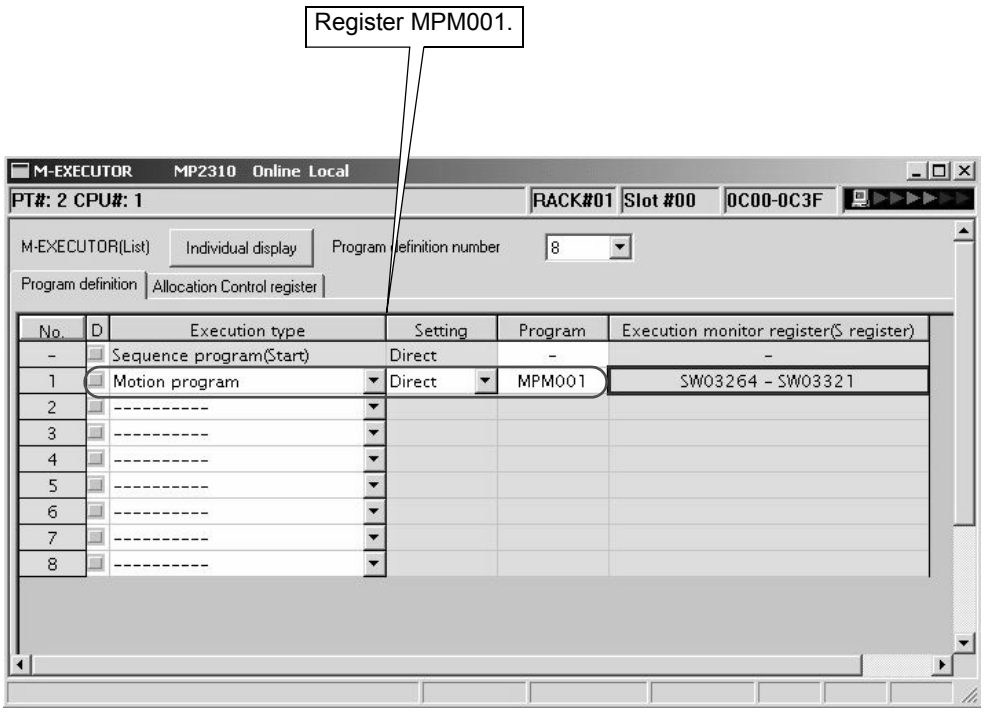
4.3.2 Registering the Program Execution

Programs can be registered in the two following ways.
The following examples shows when the motion program MPM001 is registered.

(1) Embedding an MSEE command in the Ladder Program



(2) Registering MPM001 in the M-EXECUTOR

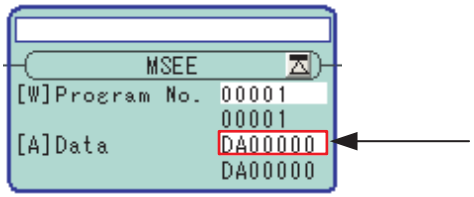


4.3.3 Work Registers

When registering motion programs as described in *4.3.2 Registering the Program Execution*, a work register to control and monitor the program execution is assigned for each registered program. The work registers are used to send instructions to the motion programs from the motion program control program, and to get the motion program status.

(1) For a Motion Program Called from the Ladder Program Using an MSEE Command

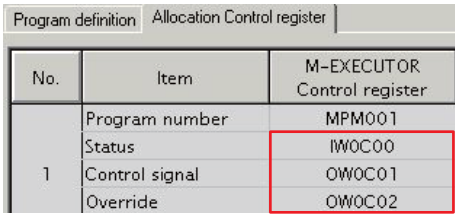
Four words of the register (MA□□□□ or DA□□□□) that are specified for Data of the MSEE command are used as the work register.



Work Register	Register No. in the Example	Contents	I/O
1st word	DW00000	Status flag	Output
2nd word	DW00001	Control signal	Input
3rd word	DW00002	Override for interpolation	Input
4th word	DW00003	System work number	Input

(2) For a Motion Program Registered in an M-EXECUTOR Program Execution Definition

The M-EXECUTOR control registers are used as the work registers.
The M-EXECUTOR control registers are automatically defined by the system.



Work Registers (M-EXECUTOR Control Registers)	Register No. in the Example	Contents	I/O
Status	IW0C00	Status flag	Output
Control signal	OW0C01	Control signal	Input
Override	OW0C02	Override for interpolation	Input

The details of the work registers are described from the next page.

(a) Status Flag

Bit No		Status	Description
0 to 3	Bit 0	Program running	This bit is ON while the motion program is running. 0: The motion program is being stopped 1: The motion program is running
	Bit 1	Program paused	This bit is ON while the motion program is paused by Program Pause Request. 0: The program is not paused by Program Pause Request. 1: The program is paused by Program Pause Request.
	Bit 2	Program stopped by stop request	This bit is ON while the motion program is stopped by Program Stop Request. 0: The program is not stopped by Program Stop Request. 1: The program is stopped by Program Stop Request.
	Bit 3	(Reserved)	–
4 to 7	Bit 4	Single program block operation stopped	This bit is ON while a single program block operation is stopped during debugging. 0: Other than single block operation stop 1: Single block operation stop
	Bit 5	(Reserved)	–
	Bit 6	(Reserved)	–
	Bit 7	(Reserved)	–
8 to B	Bit 8	Program alarm	This bit is ON while a program alarm is occurring. When this bit turns ON, the details of the error are written in the error information screen and S register. 0: No program alarm 1: A program alarm is occurring.
	Bit 9	Stopped by break point	This bit is ON while the program is stopped during a break point during debugging. 0: Not stopped at break point 1: Stopped at break point
	Bit A	(Reserved)	–
	Bit B	In debug mode	This bit is ON while debugging the motion program. 0: Normal operation mode 1: Debug mode
C to F	Bit C	Program type	Reports the running program type: Motion program or sequence program 0: Motion program 1: Sequence program
	Bit D	Start request signal history	This bit turns ON when the Program Operation Start Request is ON. 0: Program Operation Start Request OFF 1: Program Operation Start Request ON
	Bit E	No system work error Scanning error	This bit turns ON when the system work required to execute the motion program is not assigned, or when an MSEE command is embedded in a drawing other than the H drawing. 0: No system work error is not occurring. 1: No system work error is occurring.
	Bit F	Main program number exceeded error	This bit turns ON when the specified motion program number is outside the range. Motion program number range: 1 to 256 0: Within the range 1: Outside the range

(b) Control Signal

⊣ ⊢: Signals with this indication must be kept ON until they are accepted by the system.
⊣↓⊢: Signals with this indication should be ON for one high-speed scan.

Bit No		Status	Description
0 to 3	Bit 0 ⊣ ⊢ ⊣↓⊢	Program operation start request	Request to start a motion program. When this bit status changes from OFF to ON, the motion program will start running. However, this bit is disabled whenever an alarm is occurring in the motion program. 0: Program Operation Start Request OFF 1: Program Operation Start Request ON
	Bit 1 ⊣ ⊢	Program pause request	Request to pause a motion program. Resetting this bit to 0 after pausing the motion program will restart the program from the point it was stopped. 0: Program Pause Request OFF (Cancel the pause request) 1: Program Pause Request ON
	Bit 2 ⊣ ⊢	Program stop request	Request to stop a running motion program. Turning this bit ON while the axis is moving will cause a motion program alarm. 0: Program Stop Request OFF, 1: Program Stop Request ON
	Bit 3 ⊣ ⊢	Program single block mode selection	Request for program single block mode. Use this bit instead of the debugging operation 0: Program Single Block Mode Selection OFF 1: Program Single Block Mode Selection ON
4 to 7	Bit 4 ⊣ ⊢ ⊣↓⊢	Program single block start request	Changing this bit status from OFF to ON will start the program single block operation (step-by-step operation). This bit is valid when the control signal bit 3 (Program single block mode selection) is ON. 0: Program Single Block Start Request OFF 1: Program Single Block Start Request ON
	Bit 5 ⊣ ⊢	Program reset and alarm reset request	Request to reset a motion program and a alarm. 0: Program reset and Alarm Reset Request OFF, 1: Program reset and Alarm Reset Request ON
	Bit 6 ⊣ ⊢ ⊣↓⊢	Program continuous operation start request	Request to restart a program that has been stopped by the Program Stop Request, from the point it was stopped. 0: Program Continuous Operation Start Request OFF 1: Program Continuous Operation Start Request ON
	Bit 7	(Reserved)	–
8 to B	Bit 8 ⊣ ⊢	Skip 1 information	Turning this bit ON while moving the axis using an SKP command (when SS1 is selected for the skip input signal) will decelerate the axis to a stop and cancel the moving amount that remains. 0: SS1 signal OFF, 1: SS1 signal ON
	Bit 9 ⊣ ⊢	Skip 2 information	Turning this bit ON while moving the axis using an SKP command (when SS2 is selected for the skip input signal) will decelerate the axis to a stop and cancel the moving amount that remains. 0: SS2 signal OFF, 1: SS2 signal ON
	Bit A	(Reserved)	–
	Bit B	(Reserved)	–
C to F	Bit C	(Reserved)	–
	Bit D ⊣ ⊢	System work number setting	Turn this bit ON to specify the system work number. 0: Do not specify the system work number, 1: Specify the system work number
	Bit E ⊣ ⊢	Override setting for interpolation	Turn this bit ON to enable the interpolation override. 0: Disable the interpolation override. 1: Enable the interpolation override.
	Bit F	(Reserved)	

(c) Interpolation Override

Set the override value for execution of an interpolation command MVS, MCW, MCC, or SKP.
Interpolation override refers to change the output ratio of the speed reference for axis movement executed by an interpolation related command.
Interpolation override is enabled when the bit E (Override setting for interpolation) of the control signal is ON.

- Setting range of interpolation override: 0 to 32767
- Unit: 1 = 0.01%

(d) System Work Number

When using an MSEE command to call a motion program from the ladder program, set the system work number to call the motion program. The set system work number is valid when the bit D (System work number setting) is ON.

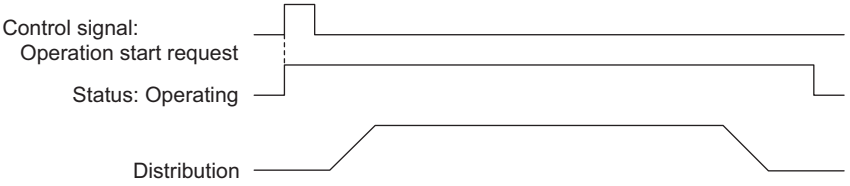
- Setting range: 1 to 16

Note: 1. When using the M-EXECUTOR, the system work number cannot be set. A system work number that is same as the definition number is used.
2. Do not specify the system work number for the M-EXECUTOR by the MSEE command when using both the MSEE command and the M-EXECUTER. Doing so will cause a no system work error.
System work number for the M-EXECUTOR: 0 to the number set in **Program definition number**.

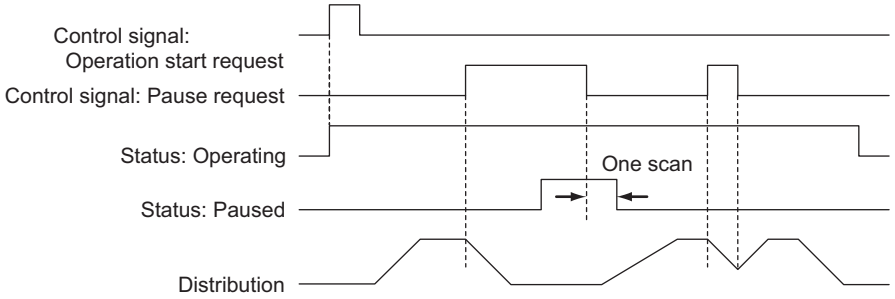
• Timing Chart for Motion Program Control Signals

The following figure shows an example of a timing chart for motion program control signals.

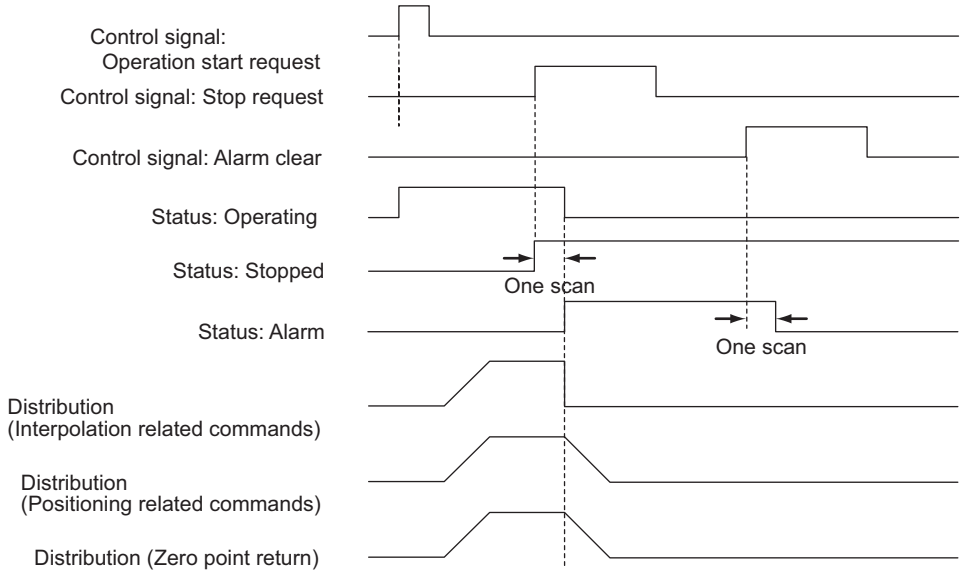
• Program Operation Start Request



• Pause Request



• Stop Request



IMPORTANT

- Turning ON the Stop Request while the axis is being moved by a motion command will cause an alarm.
- Turning ON the Stop Request while the axis is being moved by an interpolation related command will immediately stop the axis.
  Use the Pause Request to decelerate the axis to a stop.
- During execution of the Zero Point Return (ZRN) command, Pause Requests will not be accepted.
  Use the Stop Request to stop the zero point return operation.

A program example for motion program control is given in *B.1 Programs for Controlling Motion Program Execution*.

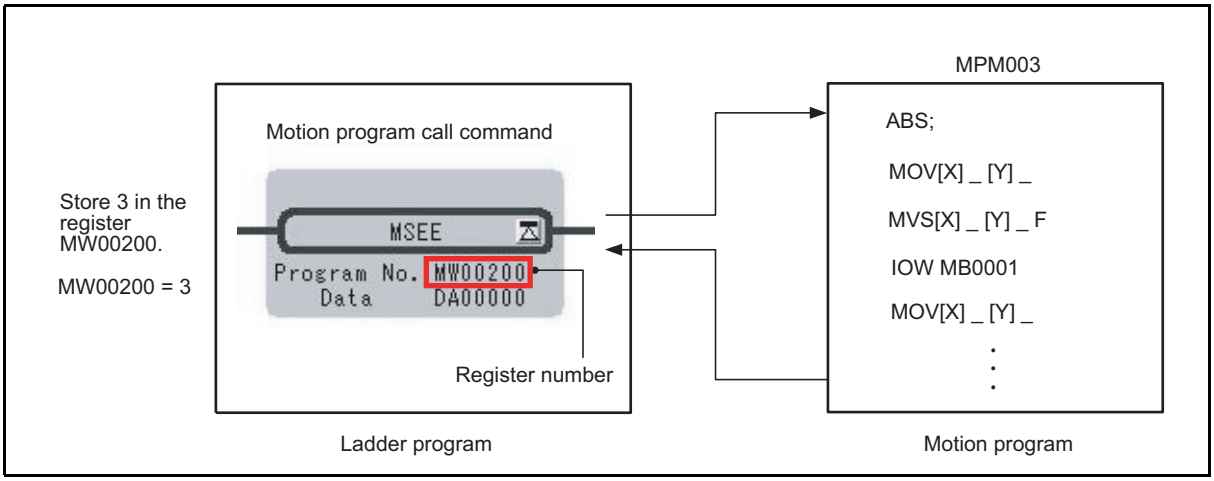
4.4 Advanced Programming

4.4.1 Indirect Designation of a Program Number Using a Register

This method calls the program (MPM□□□) that matches the value stored in the register.

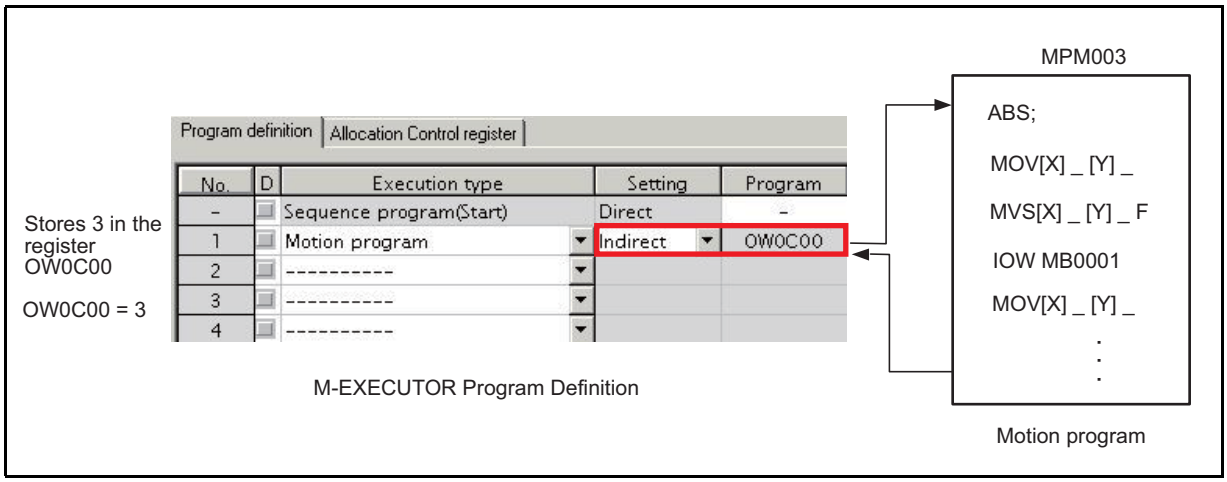
(1) When Using a Motion Program called by a MSEE Command from the Ladder Program

Specify a register (M or D register) to be used for the indirect designation in the **Program No.** of the MSEE command.



(2) For a Motion Program Registered in the M-EXECUTOR Program Execution Definition

Select **Indirect** under **Setting**. The register to be used for indirect designation will be assigned automatically by the system.



4.4.2 Controlling the Motion Program Directly from an External Device

The M-EXECUTOR Module has a function used to allocate the M-EXECUTOR's control register to a desired register.

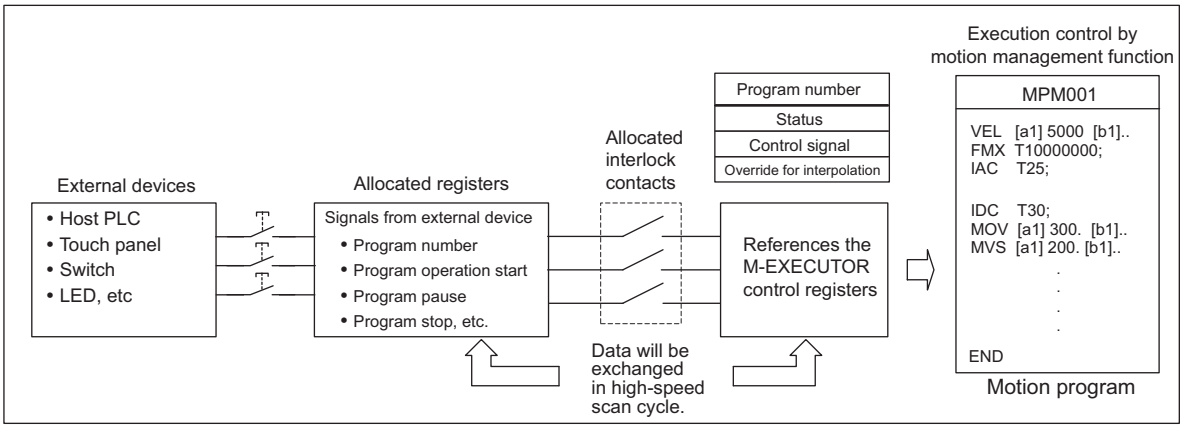
With this function, data can automatically be exchanged between the M-EXECUTOR's control register and the I/O register connected to an external device. Thus, this function allows an external device to directly control the motion program.

The following diagram shows a setting example for this function.

M-EXECUTOR Register Allocation window

No.	Item	M-EXECUTOR Control register	Allocation Disable	Direction	Allocation register	Allocation Contact interlock
1	Program number	MPM001				
	Status	IW0C00	☐	->	OW0000	IB00020
	Control signal	OW0C01	☐	<-	IW0000	IB00020
	Override	OW0C02	☐	<-	IW0001	IB00020

Set desired registers for **Allocation register** and **Allocation Contact interlock**.



INFO

■ The allocated interlock contact is used as an interlock for motion program execution. Always allocate an interlock contact in **Allocation Contact interlock** when allocating a register in **Allocation register**.

The following processes are executed according to the ON/OFF status of the allocated interlock contact.

- When the allocated interlock contact is ON, the allocated register exchanges data with the M-EXECUTOR control register in a high-speed scan cycle, and motion program execution is enabled during the data exchange.
- When the allocated interlock contact is OFF, the allocated register does not exchange data with the M-EXECUTOR control register, and motion program execution is disabled.
- When the **Allocation Contact interlock** is changed from ON to OFF while the motion program is running, the motion program will stop running and the axes will stop moving. At this moment, the motion program alarm "1Bh: Emergency Stop Command in Execution" will occur, and bit 8 (Program alarm occurring) of the status flag will turn ON.

Use the following procedure to restart the motion program.

1. Set the interlock contact from OFF to ON.
2. Set bit 5 (Program reset and alarm reset request) of the control signal to ON.
3. Confirm that bit 8 (Program alarm) of the status flag turns OFF.
4. Set bit 5 (Program reset and alarm reset request) of the control signal to OFF.
5. Set bit 0 (Program operation start request) of the control signal to ON.

4.4.3 Monitor the Motion Program Execution Information Using S Register

Using S register (SW03200 to SW04191) allows you to monitor the motion program execution information.

The way to monitor the execution information for a motion program registered in the M-EXECUTOR program execution definition differs from that for a motion program referenced by an MSEE command from a ladder program.

The way to monitor it in each case is shown as follows:

(1) A Motion Program Referenced by an MSEE Command from a Ladder Program

When a motion program is referenced by an MSEE command from a ladder program, the way differs, depending on the "Bit D" setting (system work number setting) of the motion program control signal, as follows:

(a) The motion program control signal "Bit D, System Work Number Setting" = ON

The execution information is reported to "Program Information Using Work n" register (SW03264 to SW04191). For example, when "System Work Number"=1, the motion program execution information can be monitored in SW03264 to SW03321 "Program Information Using Work 1."

(b) The motion program control signal "Bit D, System Work Number Setting" = OFF

The used system work is automatically decided by system. Thus, to check which work is used, refer to "Running Program Number" (= SW03200 to SW03215). For example, when you want to monitor the motion program MPM001 and SW03202 = 1, as used the work number=3, the execution information for the motion program MPM001 can be monitored in "Program Information Using Work 3" (=SW03380 to SW03437).

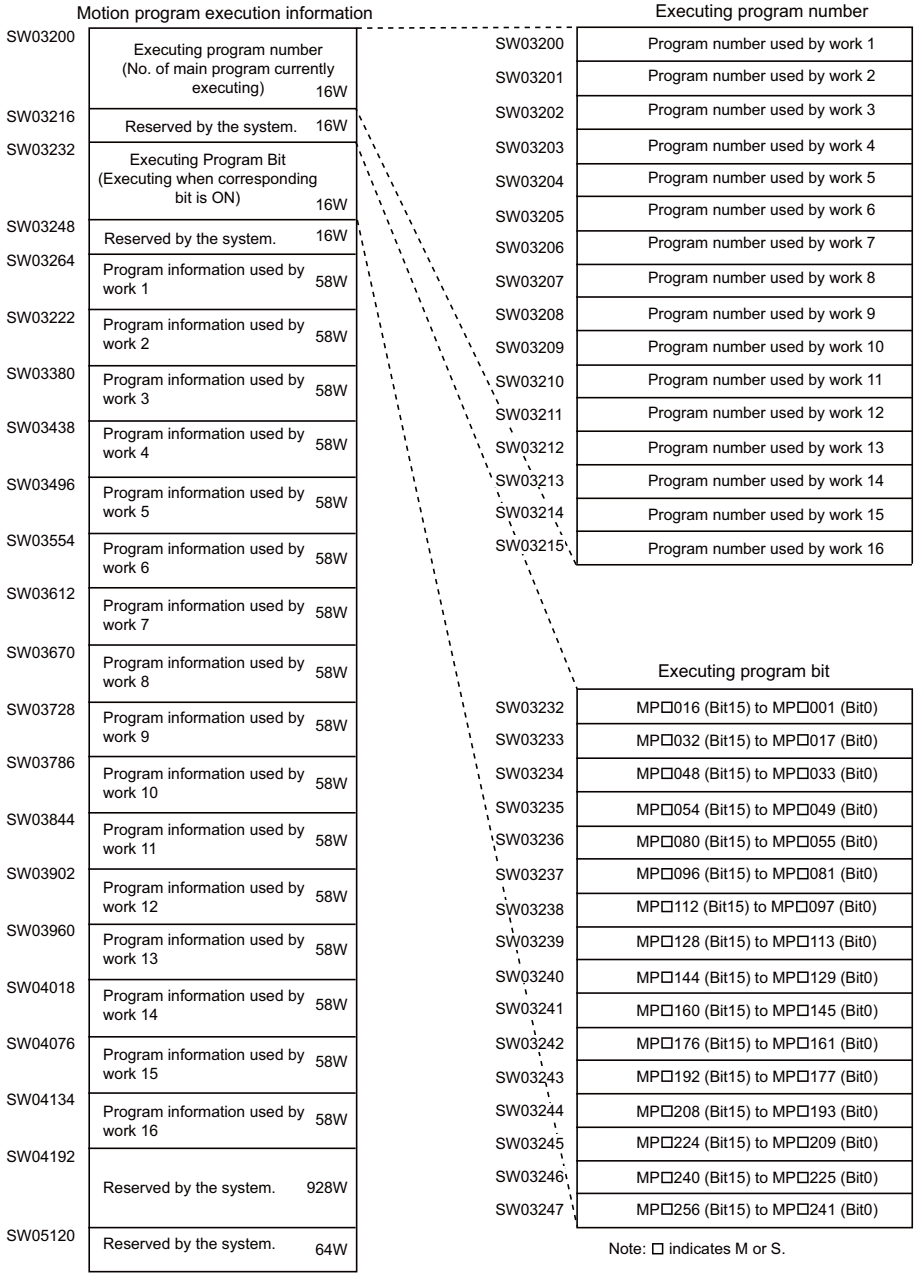
(2) A Motion Program Registered in the M-EXECUTOR Program Execution Definition

For motion programs registered in the **Program definition** tab page of the M-EXECUTOR, the system work numbers to be used are the same as for the Program execution registry number registered in the M-EXECUTOR Module.

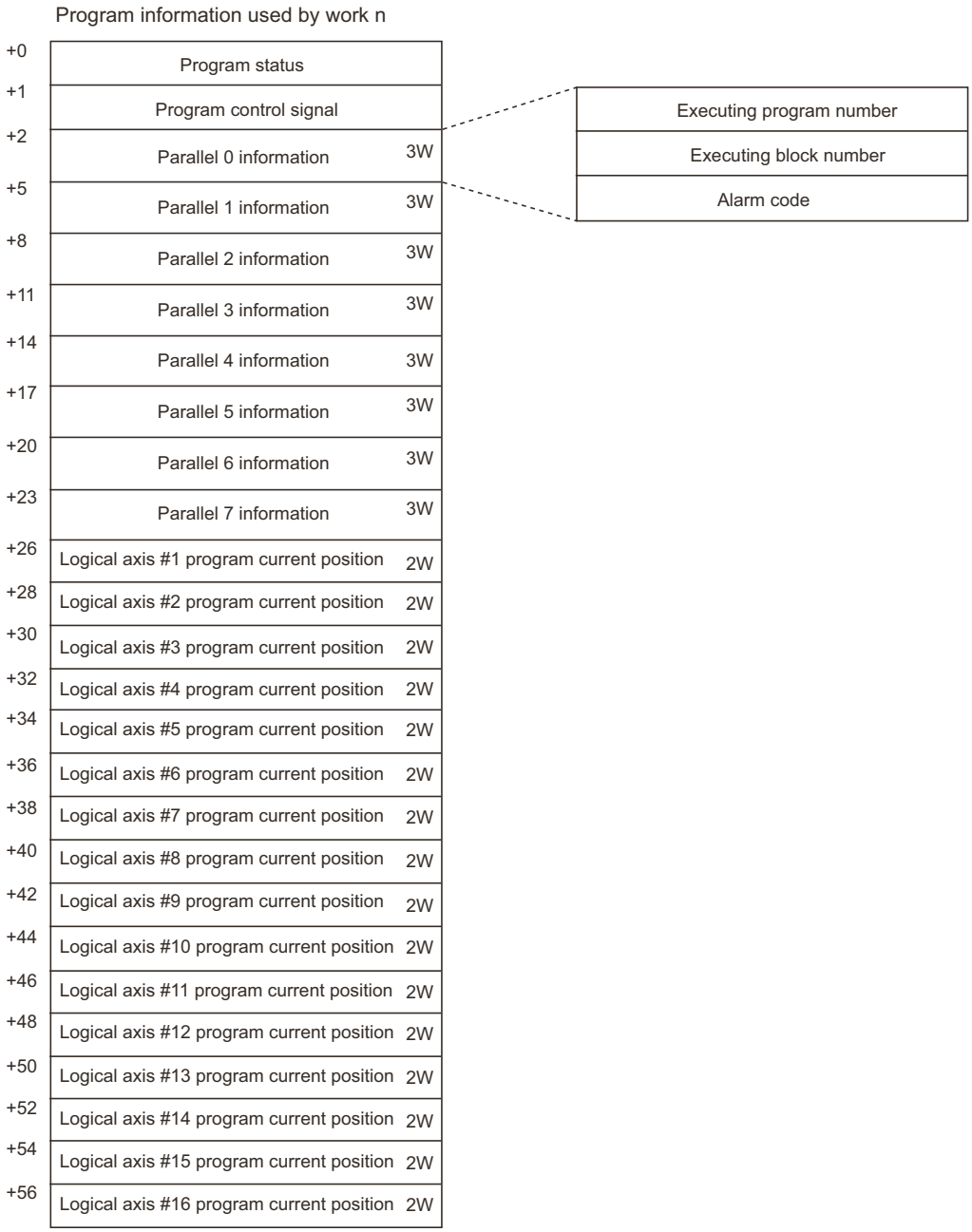
For example, a motion program is registered as "Definition No." = 3, the used system work number is "System Work" = 3. In this case, the execution information for the motion program can be monitored in "Program Information Using Work 3" (= SW03380 to SW03437).

For more information on the register area of the motion program execution information, refer to the subsequent pages.

(a) Register Areas for Motion Program Execution Information



(b) Details of Program Information Used by Work n



INFO For a list of S registers, refer *(2) Using S Register* in *10.2.3 Confirming the Alarm Code*.

5

Sequence Programs

This chapter describes sequence program types and how to run them.

5.1 Sequence Program Types ----------5-2

5.2 How to Run a Sequence Program ----------5-3

5.2.1 How to Run a Sequence Program ---------- 5-3
5.2.2 Registering Program Execution ---------- 5-4
5.2.3 Work Register ---------- 5-5

5.1 Sequence Program Types

Sequence programs are of the following two types.

Category	Designation Method	Features	Number of Programs
Main Program	SPM□□□ □□□= 1 to 256	Calling from the M-EXECUTOR program execution definition	Up to 256 programs of the following types can be created: • Main motion program • Sub motion program • Main sequence program • Sub sequence program
Subprogram	SPS□□□ □□□= 1 to 256	Calling from the main program	

INFO

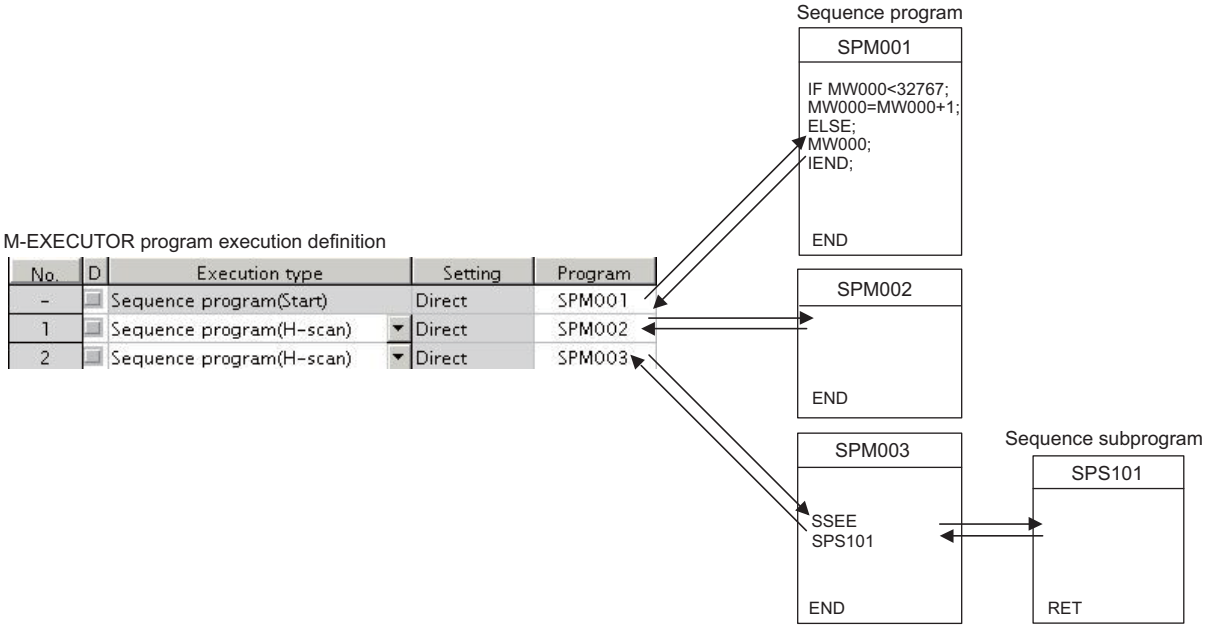
■ The program numbers of sequence programs are managed in the same manner as the motion program numbers. Assign a different number for each program number.

- Motion program MPM□□□: Program number of MPS□□□
- Sequence program SPM□□□: Program number of SPS□□□

5.2 How to Run a Sequence Program

5.2.1 How to Run a Sequence Program

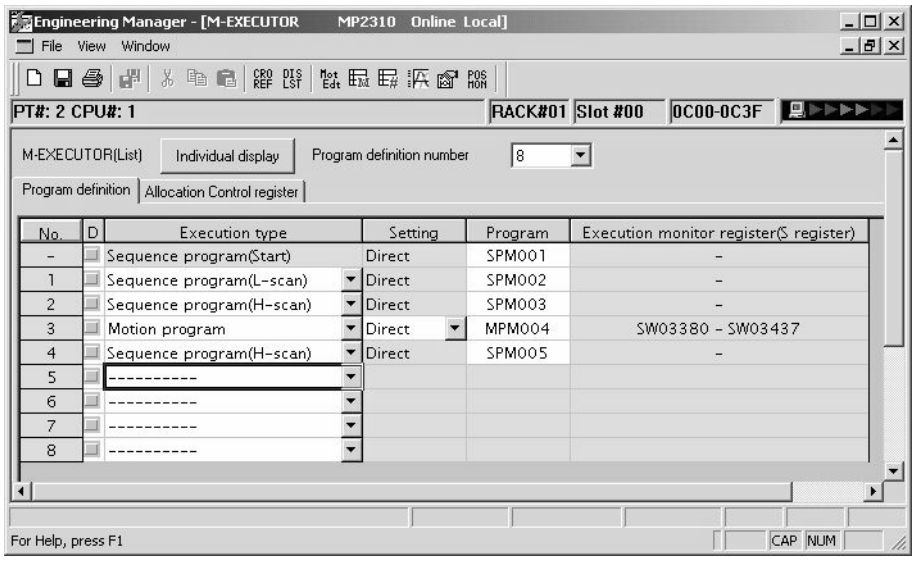
A sequence program is executed by registering it in the **Program definition** tab page of the M-EXECUTOR.



When the execution type is set to **Sequence Program (H scan)** or **Sequence Program (L scan)**, the program is executed at the time the definition is saved. When the execution type is set to **Sequence Program (Start)**, the program is executed when the power supply is turned ON again next time.

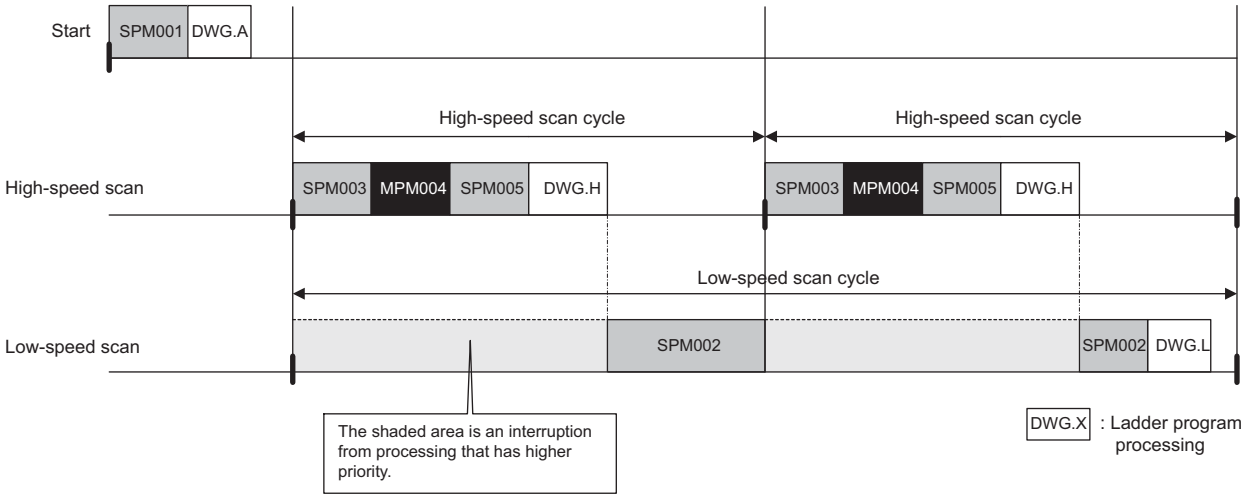
An operation example of running a sequence program is shown below.

(1) M-EXECUTOR Program Definition Tab Page



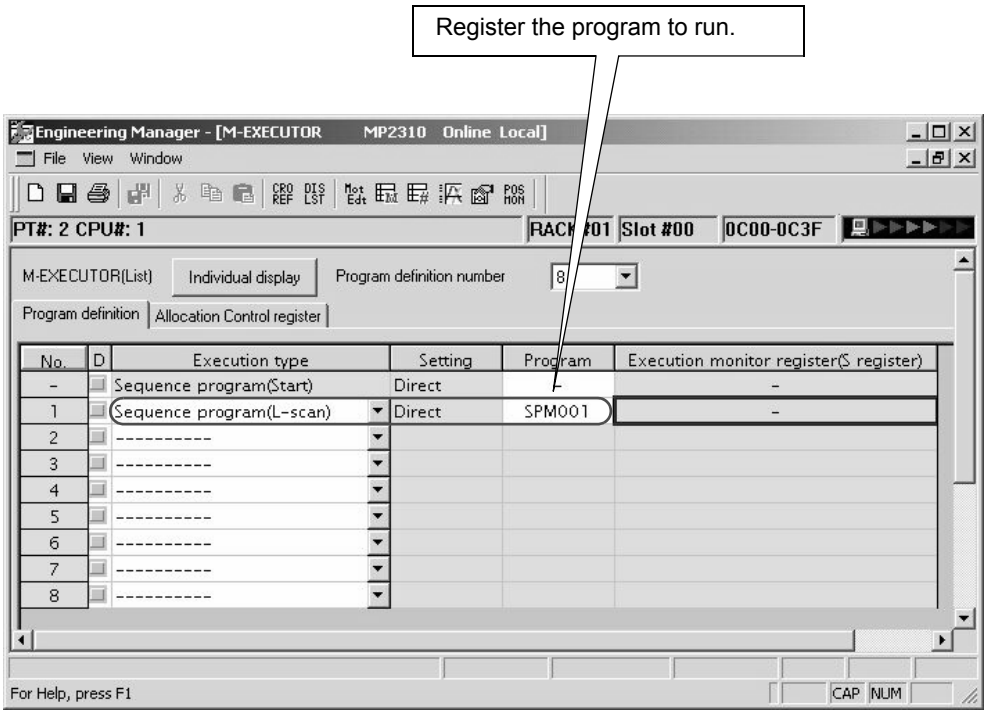
(2) Execution Timing

The following diagram illustrates program execution timing.
As shown in the following diagram, the programs are executed in the order they are registered in the **Program definition** tab page.



5.2.2 Registering Program Execution

Register the programs to run as shown below. The following screenshot shows an example of registering the sequence program SPM001 to run in a high-speed scan cycle.



INFO Only the direct designation is available for sequence programs. The indirect designation cannot be used.

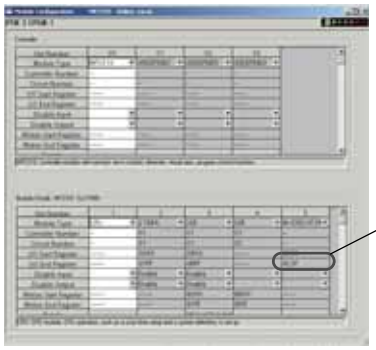
5.2.3 Work Register

A status flag to monitor program status is assigned to the sequence program in *5.2.2 Registering Program Execution*. The status flag of a sequence program can be obtained by the following equation.

IW□□□□ + 4 × (Program definition number - 1)

└ M-EXECUTOR's I/O start register number

• The I/O start register number can be confirmed in the **Module Configuration** window.



(1) Status Flag of Sequence Program

Bit No		Status	Description
0 to 3	Bit 0	Program running	This bit is ON while the sequence program is running. 0: The sequence program is being stopped, 1: The sequence program is running.
	Bit 1	(Reserved)	–
	Bit 2	(Reserved)	–
	Bit 3	(Reserved)	–
4 to 7	Bit 4	(Reserved)	–
	Bit 5	(Reserved)	–
	Bit 6	(Reserved)	–
	Bit 7	(Reserved)	–
8 to B	Bit 8	Program alarm is occurring	This bit turns ON when any of the following errors occur while calling the sequence program using an SSEE command. This bit turns OFF when the error is corrected. • The called program is not registered. • The called program is not a sequence program. • The called program is not a subprogram. (The main program is called.) • The called program number exceeded the set range. • Over nested 0: No program alarm (Normal), 1: Program alarm occurring
	Bit 9	Stopping at break point	This bit is ON when the program is stopped at a break point during debugging. 0: Not stopped at break point, 1: Stopped at break point
	Bit A	(Reserved)	–
	Bit B	In debug mode	This bit is ON while the program is running in debug mode. 0: Normal operation mode, 1: Debug mode
C to F	Bit C	Program type	Reports the running program type: Motion program or sequence program 0: Motion program, 1: Sequence program
	Bit D	Start request history	This bit is ON while the sequence program is running. 0: Sequence program stopped, 1: Sequence program running
	Bit E	(Reserved)	–
	Bit F	(Reserved)	–

6

Variables (Registers)

This chapter describes the details of variables that can be used in both motion programs and sequence programs.

6.1 Overview ----------6-2
6.1.1 Variable Types ---------- 6-2
6.1.2 Global Variables and Local Variables ---------- 6-4
6.2 Using Variables ----------6-7
6.2.1 System Variables (S Registers) ---------- 6-7
6.2.2 Data Variables (M Registers) ---------- 6-8
6.2.3 Input Variables (I Registers) ---------- 6-9
6.2.4 Output Variables (O Registers) ---------- 6-11
6.2.5 C Variables (C Registers) ---------- 6-13
6.2.6 D Variables (D Registers) ---------- 6-14
6.3 How to Use Subscripts i, j ----------6-15

6.1 Overview

This section summarizes the variables used in motion programs.

6.1.1 Variable Types

In a motion program and a sequence program, variables can be coded in place of numeric values. When variables are used in actual operations, the numeric values stored in the variable area are retrieved.

(1) Types of Variable (Registers)

The seven types of register shown in the following table can be used as variables in a motion program and a sequence program. S, M, I, O, and C registers are global variables that can be used by both motion programs and sequence programs. D registers are local variables that are defined for each program and cannot be used by other programs.

Table 6.1 Types of Variable

Type	Name	Designation Method	Range	Description	Character-istic
S	System Registers	SB,SW,SL,SFnnnnn	SW00000 to SW08191	Registers that can be referenced by the system. Register number nnnn is a decimal expression.	Common to programs
M	Data Registers	MB,MW,ML,MFnnnnn	MW00000 to MW65534	Registers common to all programs. Used as interfaces between programs. Register number nnnn is a decimal expression.	
I	Input Registers	IB,IW,IL,IFhhhh	IW0000 to IW7FFF	Registers used for input data. Register number hhhh is a hexadecimal expression. The register numbers 8000 and onward are used as motion monitoring parameters.	
O	Output Registers	OB,OW,OL,OFhhhh	OW0000 to OW7FFF	Registers used for output data. Register number hhhh is a hexadecimal expression. The register numbers 8000 and onward are used as motion setting parameters.	
C	Constant Registers	CB,CW,CL,CFnnnnn	CW00000 to CW16383	Registers that can be referenced only by a program. Register number nnnn is a decimal expression.	
D	D Registers	DB,DW,DL,DFnnnnn	DW00000 to DW16383	Internal registers unique to each program. Can only be used by the corresponding program. The actual range to be used is specified by the user on the MPE720. Register number nnnn is a decimal expression.	Unique to a program

IMPORTANT

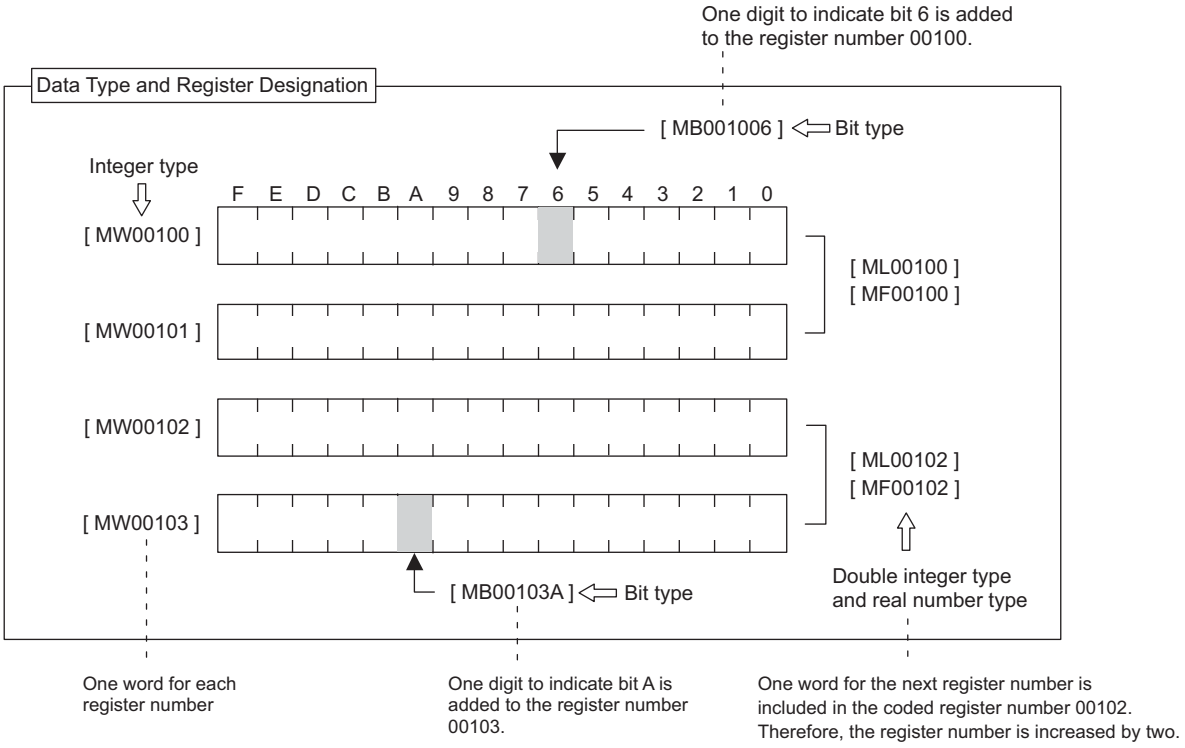
The # registers cannot be used in motion programs or sequence programs. If a # register is used in a motion program or sequence program, a syntax error will occur when saving the program.

(2) Data Types

As shown in the following table, the data types are bit, integer, double integer, and real number data. Use them as required.

Table 6.2 Data Types

Symbol	Data Type	Numeric Range	Remarks
B	Bit	ON (1), OFF (0)	Used to determine the relay sequence and the ON/OFF condition.
W	Integer	-32768 to +32767 (8000H to 7FFFH)	Used for arithmetic operations. The parentheses ( ) show its use in logic operations.
L	Double Integer	-2147483648 to +2147483647 (80000000H to 7FFFFFFFH)	Used for arithmetic operations. The parentheses ( ) show its use in logic operations.
F	Real Number	± (1.175E-38 to 3.402E + 38)	Used for high-level arithmetic operations.



6.1.2 Global Variables and Local Variables

(1) Global Variables

Global variables can be used in common by ladder logic programs, user functions, and the drawings in motion programs and sequence programs. In other words, the calculated results for a given ladder logic program can be used by other user functions and motion programs. The global variable size is stored by the system for each variable. (See the following illustration.)

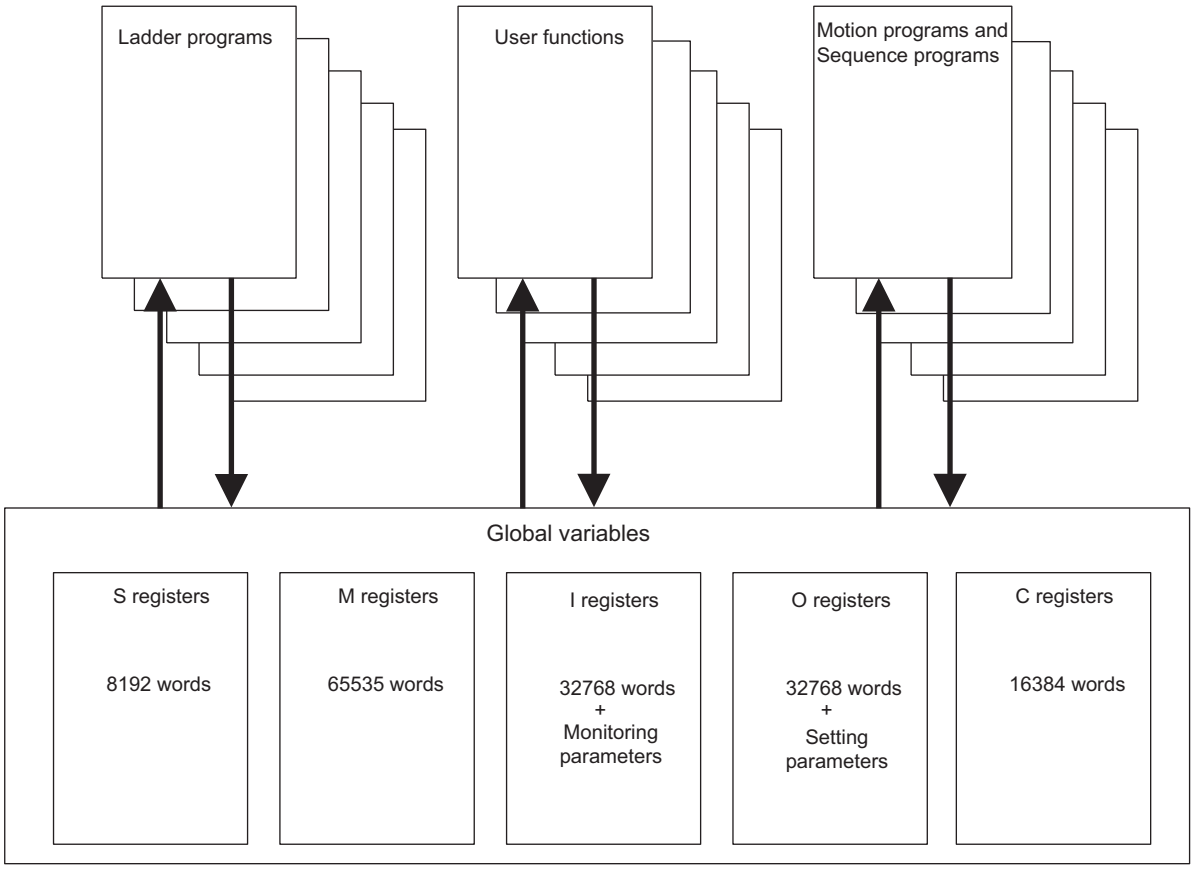
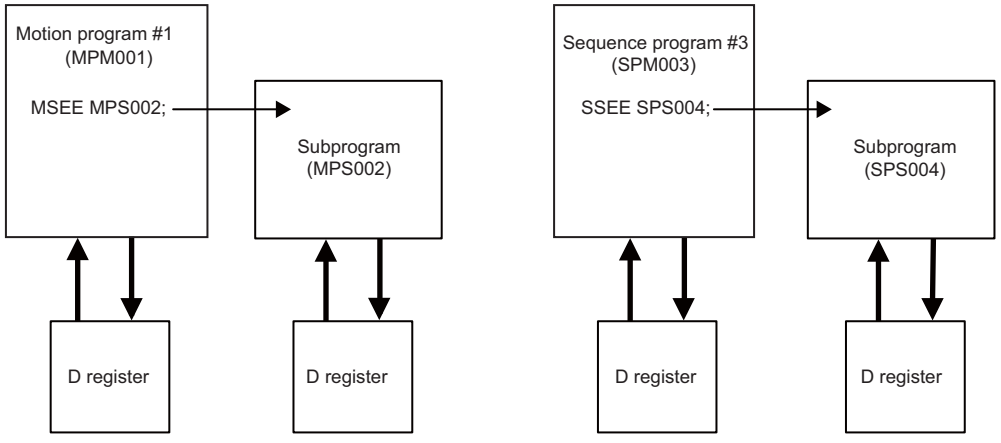


Fig. 6.1 Global Variables

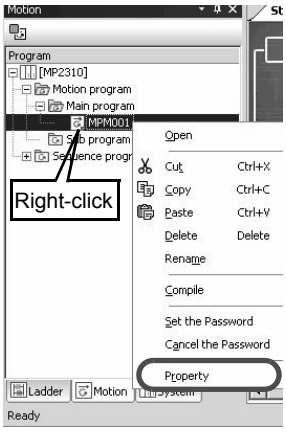
(2) Local Variables

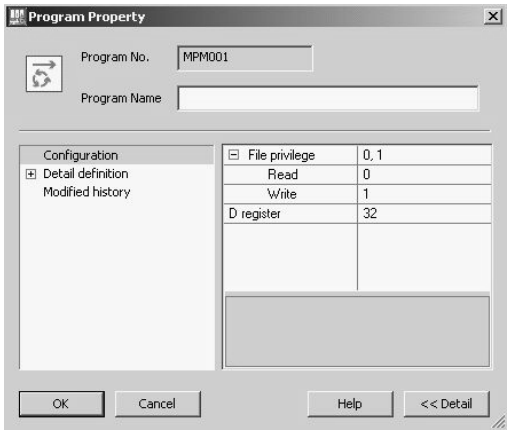
Local variables are used locally by each program. They cannot be used by other programs. Local variables are stored in the corresponding program memory.



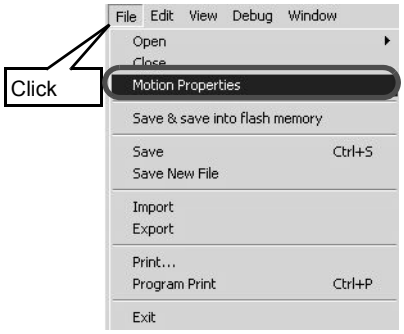
The number of local variables (D registers) to be used in each program can be specified in the **Program Property** window or **Motion Program Configuration Definition** window. Up to 16384 words can be used for one drawing.

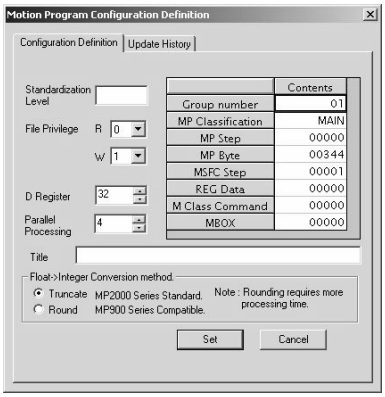
Program Property window





Motion Program Configuration Definition window





INFO # registers cannot be used in motion programs or sequence programs.

IMPORTANT

■ Precautions for Variable Operations

Storing data in a variable of a different data type will result in:

Format	• Use a Substitute command (=). • Write a destination register on the left and the operation on the right. MW00100 = MW00101 + MW00102;
Variable Operations	• When real number data is stored in an integer type variable. MW00100 = MF00200; The real number data is converted into integer (00001) (1.234) data and stored in the destination register. Note: Be careful to avoid a round-off error, caused when storing real number data into an integer type variable. Specify the round-off method when storing the real number in an integer type variable in the **Motion Program Configuration Definition** window. MW00100 = MF00200 + MF00202; (0124) (123.48) (0.02) The operation result will differ depending (0123) (123.49) (0.01) on the variable values to be calculated. • When real number data is stored in a double integer type variable. ML00100 = MF00200; The real number data is converted to integer (65432) (65432.1) data and stored in the destination register. • When double integer data is stored in an integer type variable. MW00100 = ML00200; The lower 16 bits of double integer data are (-00001) (65535) stored in the destination register as they are. • When integer data is stored in a double integer type variable. ML00100 = MW00200; The integer data is converted to double integer (0001234) (1234) data and stored in the destination register.
Operations that Cause Syntax Errors	• When integer data is stored in a bit type variable, MB000100 = 123; => Syntax error MB000100 = MW00100; => Syntax error

6.2 Using Variables

This section explains how to use variables.

6.2.1 System Variables (S Registers)

(1) Overview

System variables (S registers) are provided by the MP2000-series Machine Controller system. They can be used to read system error information, the operation status, and so on. S registers are global variables that can be used in any motion program and sequence program. For details, refer to the user's manual for the Machine Controller to be used.

(2) Description

S registers are designated as follows:

```
SB000000 to SB08191F
SW00000 to SW08191
SL00000 to SL08190
SF00000 to SF08190
```

The variable number is expressed as a decimal. When bits are specified, the bit number is expressed in hexadecimal.

(3) Programming Examples

◀ EXAMPLE ▶

- Bit Designation

```
OB000010 = SB000402|SB000403;
```

- Integer Designation

```
MW00100 = SW00041;
```

- Double Integer Designation

```
ML00100 = SL00062;
```

IMPORTANT

The system registers (S) are used exclusively for reading. If they are written to, system operations cannot be guaranteed.

6.2.2 Data Variables (M Registers)

(1) Overview

M registers are general-purpose variables that can be used in ladder logic programs, user functions, motion programs, and sequence programs. These are global variables that can be used as interfaces between motion programs, sequence programs, and ladder logic programs.

(2) Description

M registers are designated as follows:

```
MB000000 to MB65534F
MW00000 to MW65534
ML00000 to ML65533
MF00000 to MF65533
```

The M register can be used as a variable for each type of operation and substituted for the operation result, or specified as the variable for the positioning coordinate value or the speed. The variable number is expressed as a decimal.

(3) Programming Examples

(a) Specifying the Position and Speed in Axis Move Commands as Variables

◀ EXAMPLE ▶

```
• Parameter Reference unit = mm
            When decimal point position = 3
    ML00100=100000;          → 100.000 mm
    ML00102=200000;          → 200.000 mm
    ML00104=300000;          → 300.000 mm
    ML00106=500000;          → 500.000 mm/min
    MVS [X]ML00100 [Y]ML00102 [Z]ML00104 FML00106;
```

(b) Using Variables in Operations

◀ EXAMPLE ▶

- Bit Designation

```
MB001001=IB00100 & IB00201;
```

- Integer Designation

```
MW00101=(MW00101 | MW00102) & FF0CH;
```

- Double Integer Designation

```
ML00200=((MI00202*ML00204) / ML00206)*3;
```

- Real Number Designation

```
MF00200=((MF00202*MF00204) / MF00206)*3.14;
```

IMPORTANT

When the travel distance coordinate values or speed is designated as a variable in the following motion commands, double integer data must be used.
MOV, MVS, MCW/MCC, ZRN, SKP, MVT, EXM, POS, ACC, SCC, IAC, IDC, IFP, FMX, INP, VEL

6.2.3 Input Variables (I Registers)

(1) Overview

These variables are used by input data and the servo monitor parameters. Although servo parameters can also be used for writing data, the values can not be guaranteed.

(2) Description

I registers are designated as follows:

IW0000 to IW7FFF: Input data IW8000 to IWFFFF: Motion monitoring parameters

(a) Register Numbers of Input Data

Depends on the address specified in the module configuration definition.

(b) Register Numbers of Motion Monitor Parameter

The number of controlled axes depends on the module type. The following indicates the number of controlled axes for each module and the maximum number of modules.

Table 6.3 Number of Axes Controlled by One Module

Motion Module	Number of Axes Controlled by One Module	Max. Number of Modules That Can Be Mounted on One Machine Controller									
		MP2100	MP2100M	MP2200	MP2300	MP2300S	MP2310	MP2400	MP2500(D)	MP2500M	MP2500MD
MP2000 Series Built-in SVB	16 max. *1	1	1	–	1	1	1	1	1	1	1
MP2100M/MP2500M SVB-01	16 max. *1	–	1	–	–	–	–	–	–	1	1
Optional Module SVB-01	16 max. *1	–	14 *2 *3	16 *3	2	1	3	–	–	–	14 *2 *3
Optional Module SVA-01	2										
Optional Module PO-01	4										

* 1. The number of axes controlled by one built-in SVB or SVB-01 depends on the MECHATROLINK definitions.
* 2. With MP2100M and MP2500MD, an expansion rack is required to use optional modules (SVB-01, SVA-01, and PO-01).
* 3. The maximum number of optional modules (SVB-01, SVA-01, and PO-01) that can be mounted on MP2100M, MP2200, or MP2500MD with an expansion rack.

Table 6.4 Register Numbers of Motion Parameter

Axis No. Circuit No.	Axis 1	Axis 2	Axis 3	Axis 4	Axis 5	••	Axis 16
1	8000 to 807F	8080 to 80FF	8100 to 817F	8180 to 81FF	8200 to 827F	••	8780 to 87FF
2	8800 to 887F	8880 to 88FF	8900 to 897F	8980 to 89FF	8A00 to 8A7F	••	8F80 to 8FFF
3	9000 to 907F	9080 to 90FF	9100 to 917F	9180 to 91FF	9200 to 9A7F	••	9780 to 97FF
4	9800 to 987F	9880 to 98FF	9900 to 997F	9980 to 99FF	9A00 to 997F	••	9F80 to 9FFF
5	A000 to A07F	A080 to A0FF	A100 to A17F	A180 to A1FF	A200 to A27F	••	A780 to A7FF
6	A800 to A87F	A880 to A8FF	A800 to A87F	A980 to A9FF	AA00 to AA7F	••	AF80 to AFFF
7	B000 to B07F	B080 to B0FF	B100 to B17F	B180 to B1FF	B200 to B27F	••	B780 to B7FF
8	B800 to B87F	B880 to B8FF	B900 to B97F	B980 to B9FF	BA00 to BA7F	••	BF80 to BFFF
9	C000 to C07F	C080 to C0FF	C100 to C17F	C180 to C1FF	C200 to C27F	••	C780 to C7FF
10	C800 to C87F	C880 to C8FF	C900 to C97F	C980 to C9FF	CA00 to CA7F	••	CF80 to CFFF
11	D000 to D07F	D080 to D0FF	D100 to D17F	D180 to D1FF	D200 to D27F	••	D780 to D7FF
12	D800 to D87F	D880 to D8FF	D900 to D97F	D980 to D9FF	DA00 to DA7F	••	DF80 to DFFF
13	E000 to E07F	E080 to E0FF	E100 to E17F	E180 to E1FF	E200 to E27F	••	E780 to E7FF
14	E800 to E87F	E880 to E8FF	E900 to E97F	E980 to E9FF	EA00 to A97F	••	EF80 to EFFF
15	F000 to F07F	F080 to F0FF	F100 to F17F	F180 to F1FF	F200 to F27F	••	F780 to EFFF
16	F800 to F87F	F880 to F8FF	F900 to F97F	F980 to F9FF	F900 to F97F	••	FF80 to FFFF

↑
Module number offset

INFO

The register numbers for each axis of the motion monitor parameters can be obtained by the following equation.

The register start number of the motion monitor parameter = IW8000 + (Circuit No. -1) × 800h + (Axis No. -1) × 80h

(3) Programming Examples

The Input data and servo monitor parameters are read out and referenced.

◀ EXAMPLE ▶

- Bit Designation

```
MB01000 = IB0010 & IB00105;
```

- Integer Designation

```
MW0100 = IW8008;
```

- Double Integer Designation

```
ML0100 = IL8004;
```

6.2.4 Output Variables (O Registers)

(1) Overview

These variables are used for output data and servo setting parameters.

(2) Description

O registers are designated as follows:

OW0000 to OW7FFF: Output data OW8000 to OWFFFF: Motion setting parameter

(a) Register Numbers of Output Data

Depends on the address specified in the module configuration definition.

(b) Register Numbers of Motion Setting Parameter

The number of controlled axes depends on the module type. The following indicates the number of controlled axes for each module and the maximum number of modules.

Table 6.5 Number of Axes Controlled by One Module

Motion Module	Number of Axes Controlled by One Module	Max. Number of Modules That Can Be Mounted on One Machine Controller									
		MP2100	MP2100M	MP2200	MP2300	MP2300S	MP2310	MP2400	MP2500(D)	MP2500M	MP2500MD
MP2000 Series Built-in SVB	16 max. *1	1	1	–	1	1	1	1	1	1	1
MP2100M/MP2500M SVB-01	16 max. *1	–	1	–	–	–	–	–	–	1	1
Optional Module SVB-01	16 max. *1	–	14 *2 *3	16 *3	2	1	3	–	–	–	14 *2 *3
Optional Module SVA-01	2										
Optional Module PO-01	4										

* 1. The number of axes controlled by one built-in SVB or SVB-01 depends on the MECHATROLINK definitions.
* 2. With MP2100M and MP2500MD, an expansion rack is required to use optional modules (SVB-01, SVA-01, and PO-01).
* 3. The maximum number of optional modules (SVB-01, SVA-01, and PO-01) that can be mounted on MP2100M, MP2200, or MP2500MD with an expansion rack.

Table 6.6 Register Numbers of Motion Parameter

Axis No. Circuit No.	Axis 1	Axis 2	Axis 3	Axis 4	Axis 5	••	Axis 16
1	8000 to 807F	8080 to 80FF	8100 to 817F	8180 to 81FF	8200 to 827F	••	8780 to 87FF
2	8800 to 887F	8880 to 88FF	8900 to 897F	8980 to 89FF	8A00 to 8A7F	••	8F80 to 8FFF
3	9000 to 907F	9080 to 90FF	9100 to 917F	9180 to 91FF	9200 to 9A7F	••	9780 to 97FF
4	9800 to 987F	9880 to 98FF	9900 to 997F	9980 to 99FF	9A00 to 997F	••	9F80 to 9FFF
5	A000 to A07F	A080 to A0FF	A100 to A17F	A180 to A1FF	A200 to A27F	••	A780 to A7FF
6	A800 to A87F	A880 to A8FF	A800 to A87F	A980 to A9FF	AA00 to AA7F	••	AF80 to AFFF
7	B000 to B07F	B080 to B0FF	B100 to B17F	B180 to B1FF	B200 to B27F	••	B780 to B7FF
8	B800 to B87F	B880 to B8FF	B900 to B97F	B980 to B9FF	BA00 to BA7F	••	BF80 to BFFF
9	C000 to C07F	C080 to C0FF	C100 to C17F	C180 to C1FF	C200 to C27F	••	C780 to C7FF
10	C800 to C87F	C880 to C8FF	C900 to C97F	C980 to C9FF	CA00 to CA7F	••	CF80 to CFFF
11	D000 to D07F	D080 to D0FF	D100 to D17F	D180 to D1FF	D200 to D27F	••	D780 to D7FF
12	D800 to D87F	D880 to D8FF	D900 to D97F	D980 to D9FF	DA00 to DA7F	••	DF80 to DFFF
13	E000 to E07F	E080 to E0FF	E100 to E17F	E180 to E1FF	E200 to E27F	••	E780 to E7FF
14	E800 to E87F	E880 to E8FF	E900 to E97F	E980 to E9FF	EA00 to A97F	••	EF80 to EFFF
15	F000 to F07F	F080 to F0FF	F100 to F17F	F180 to F1FF	F200 to F27F	••	F780 to EFFF
16	F800 to F87F	F880 to F8FF	F900 to F97F	F980 to F9FF	F900 to F97F	••	FF80 to FFFF

↑
Module number offset

INFO

The register numbers for each axis of motion setting parameters can be obtained by the following equation.

The register start number of the motion setting parameter
= OW8000 + (Circuit No. -1) × 800h + (Axis No. -1) × 80h

(3) Programming Examples

The output data and motion setting parameters are written in.

◀ EXAMPLE ▶

- Bit Designation

```
OB01000=MB001000 & IB00100;
```

- Integer Designation

```
OW8008=MW00100;
```

- Double Integer Designation

```
OL8010=ML00100+ML00200;
```

6.2.5 C Variables (C Registers)

(1) Overview

C registers are variables to be referenced from programs. They cannot be used to write.

(2) Description

C registers are designated as follows:

```
CW00000 to CW16383
```

C registers cannot be written from programs.

(3) Programming Examples

◀ EXAMPLE ▶

Using Variables in Operations.

- Bit Designation

```
MB001000=CB001001;
```

- Integer Designation

```
MW00100=CW00100;
```

- Double Integer Designation

```
ML00100=CL00100;
```

- Real Number Designation

```
MF00100=CF00100;
```

6.2.6 D Variables (D Registers)

(1) Overview

D variables can be used only by the relevant program using specific internal registers for each motion program and sequence program.

(2) Description

D registers are designated as follows:

```
DW00000 to DW16383 (Maximum)
```

The D register can be used as a variable for each type of operation and substituted for the operation result, or specified as the variable for the positioning coordinate value or the speed. The variable number is expressed as a decimal.

The size is specified in the program configuration definition (Motion Properties), and the default is 32words max.

(3) Programming Examples

(a) Specifying the Position and Speed in Axis Move Commands as Variables

◀ EXAMPLE ▶

```
• Parameter     Reference unit = mm
                When decimal point position = 3
    DL00100=100000;                                          → 100.000 mm
    DL00102=200000;                                          → 200.000 mm
    DL00104=300000;                                          → 300.000 mm
    DL00106=500000;                                          → 500.000 mm/min
    MVS [A1]DL00100 [B1]DL00102 [C1]DL00104 FDL00106;
```

(b) Using Variables in Operations

◀ EXAMPLE ▶

- Bit Designation

```
DB001000=IB01001 & MB000101;
```

- Integer Designation

```
DW00102= (CW00103 | DW00104) & DW00105;
```

- Double Integer Designation

```
DL00106= (DL00108*ML00011) / ML00200;
```

- Real Number Designation

```
DF00200= (MF00202*DF00202)*3.14;
```

IMPORTANT

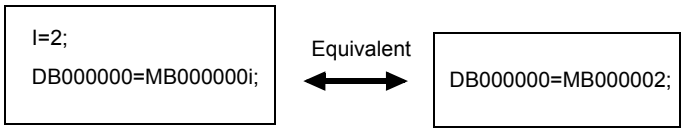
When the travel distance coordinate value or speed is designated as a variable in the following motion commands, double integer data must be used.

MOV, MVS, MCW/MCC, ZRN, SKP, MVT, EXM, POS, ACC, DCC, SCC, IAC, IDC, IFP, FMX, INP, VEL

6.3 How to Use Subscripts i, j

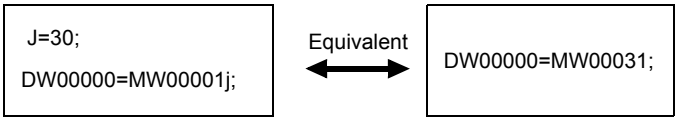
Two type of registers (i, j) are available as dedicated registers to modify the relay and register numbers. Both i and j have the same function. They are used when you want to handle a register number as a variable.
An example for each register data type is given as explanation.

(1) Bit Type Attached with a Subscript



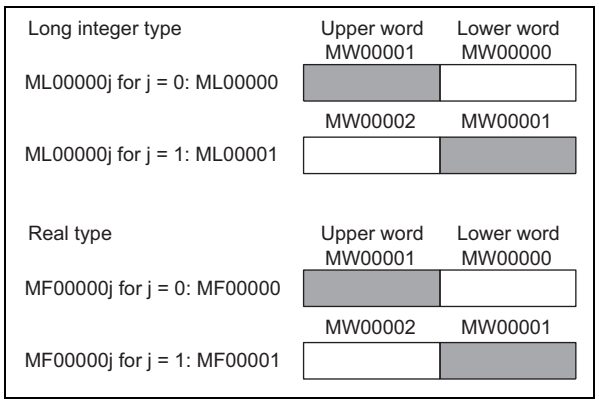
The result is a register number added with i or j value.
For example, MB000000i for i=2 is the same as MB000002. Also, MB000000j for j=27 is the same as MB00001B.

(2) Integer Type Attached with a Subscript



The result is a register number added with i or j value.
For example, MW00010i for i=3 is the same as MW00013. Also, MW00001j for j=30 is the same as MW00031.

(3) Long Integer or Real Type Attached with a Subscript



The result is a register number with an added i or j value.
For example, "ML00000j for j=1" is the same as ML00001. Also, "MF00000j for j=1" is the same as MF00001.
For double integer registers and real number registers, one word of register number and one word of one added register number are used. Be careful to avoid overlapping one with the other when using subscripts. For ML00000j when j = 0 and ML00000j when j = 1, for example, one word of MW00001 will be overlapped.

Program example using subscript is as follows.

◀ EXAMPLE ▶

```
    :
    :
ML00200 = 0 ;
J = 0 ;
WHILE J < 100 ;
   ML00200 = ML00200 + ML00100j ;
   J = J + 2 ;
WEND ;
   :
  :
```

The left program uses a subscript j and calculates the total amount of 50 registers from ML00100 to ML00198, and stores the total amount in ML00200.



- The following versions of system software and programming tool MPE720 are required to use subscripts i and j.

MP2000 Series Machine Controller	Required Version
All models	Ver.2.63 or later

MPE720	Required Version
MPE720 Ver.5	Ver.5.41 or later
MPE720 Ver.6	Ver.6.06 or later Ver.6.06 Lite or later

- Both upper and lower case letters can be used for subscripts i and j.

```
i = 0;
I = 0;
DW00000 = MW00000i ;
DW00000 = MW00000I ;
```

7

Programming

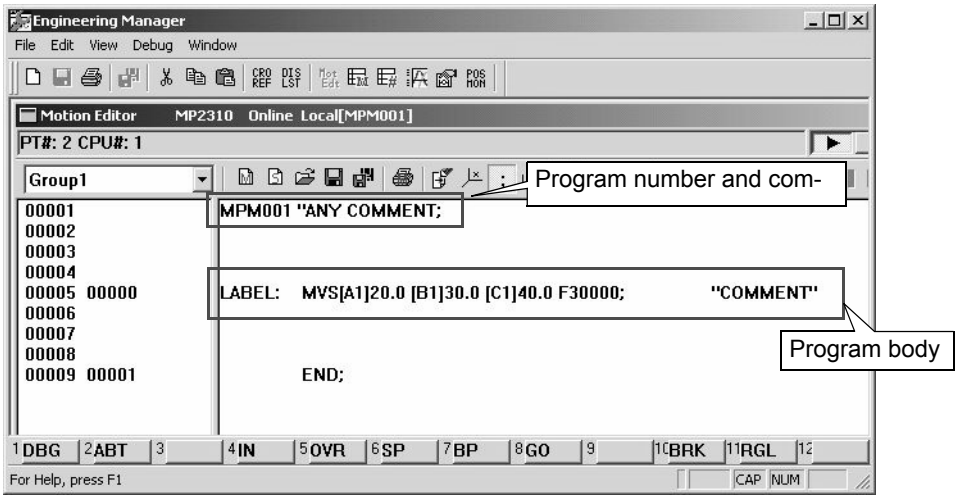
This chapter describes the rules for creating motion programs and sequence programs.

7.1 Motion Program Format ----------7-2
7.1.1 Motion Program Structure ---------- 7-2
7.1.2 Block Format ---------- 7-2
7.1.3 Using Constants and Variables ---------- 7-7
7.2 Motion Module Parameters ----------7-9
7.2.1 Axis Type Selection ---------- 7-9
7.2.2 Reference Unit ---------- 7-9
7.2.3 Electronic Gear ---------- 7-10
7.2.4 Speed Reference ---------- 7-12
7.2.5 Acceleration/Deceleration Setting ---------- 7-12
7.3 Group Definition ----------7-13
7.4 Priority Levels of Operations ----------7-15
7.5 Commands and Execution Scans ----------7-17
7.5.1 Command Types ---------- 7-17
7.5.2 List of Command Types ---------- 7-18
7.6 Sequence Program Format ----------7-19

7.1 Motion Program Format

7.1.1 Motion Program Structure

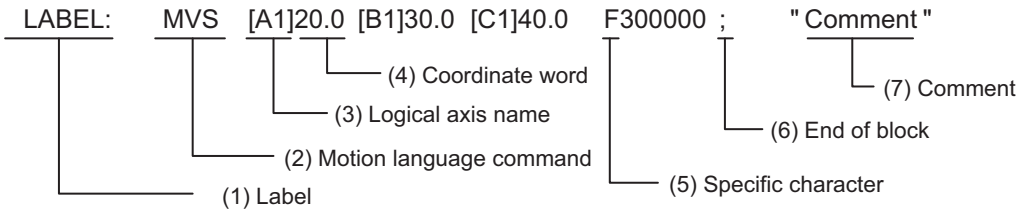
A motion program contains a program number, arbitrary comment, program body, and END command. The process executed by a motion program is coded in the program body. The motion program basic structure is shown below.



INFO One line of program number and comment can be omitted.

7.1.2 Block Format

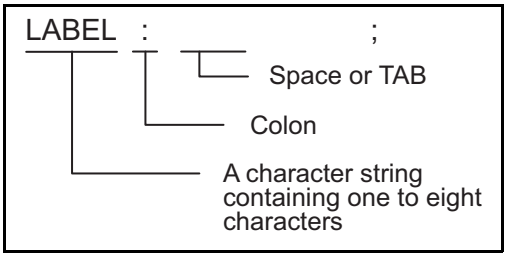
One block is one process execution unit. The program body consists of one or more than one block. The format of one block is shown below. The details of each item are described from the next page.



No.	Items	Description
(1)	Label	Indicates the destination of PFROK and SFORK command.
(2)	Motion language command	Specifies a motion program command.
(3)	Logical axis name	Specifies a logical axis name set in the **Group Definition** window.
(4)	Coordinate words	Specifies the axis coordinate value or the incremental travel amount of axis.
(5)	Specific character	Specifies supplemental data for the motion command.
(6)	End of block	Specifies the end of a block.
(7)	Comment	Describes the program comment.

(1) Label

A label consists of a character string containing from one to eight alphanumeric characters or symbols, a colon [:], and a space or TAB.



Type	Available Characters
Letters	A to Z, a to z
Numbers	0 to 9
Symbols	_ (hyphen)

Note: Numbers cannot be used at the heads of labels.

A label is required when using a Parallel Fork command (PFORK) or Selective Fork command (SFORK) command. If a PFORK or SFORK command is not used, the label description is not required.

◀ EXAMPLE ▶

Label Coding Example

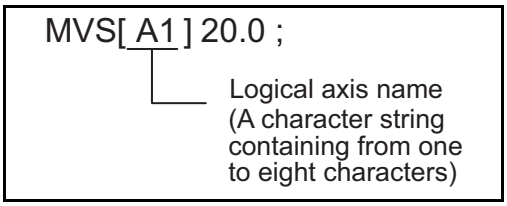
```
PFORK LAB1, LAB2;
LAB1: ZRN [A1]0 [B1]0 [C1]0;
JOINTO LAB3;
LAB2: MVS [D1]100.0 [E1]200.0 [F1]300.0;
JOINTO LAB3;
LAB3: PJOINT;
```

(2) Motion Language Commands

Code a motion language command.
Refer to *Chapter 8 Command Reference* or *Appendix A Motion Language Commands* for information on motion language commands.

(3) Logical Axis Name

Code a logical axis name set in the **Group Definition** window. The logical axis name must be enclosed in brackets [ ].



Type	Available Characters
Letters	A to Z, a to z
Numbers	0 to 9

(4) Coordinate Words

A coordinate word is a numerical value or a variable to be coded after an axis name. A coordinate word specifies the reference position, speed, acceleration/deceleration rate, and other items.

- Using Numbers for a Coordinate Word

  Write a numerical value after an axis name to directly specify the coordinate word.

  Both integers and real numbers can be used for a numerical value. However, special care must be taken when using integers.

  When the reference unit is set to 0.001 mm and the reference position "1000 (integer numbers)" is written in the coordinate word, for example, the Machine Controller interprets it as 1.000 mm. When writing "1.000 (real numbers)," the Machine Controller interprets it as it is (1.000 mm).

◀ EXAMPLE ▶

```
MVS  [A1]1000;      → 1.000 mm
or
MVS  [A1]1.000;     → 1.000 mm
or
MVS  [A1]1.;        → 1.000 mm
```

- Using a Variable for a Coordinate Word

  Write a double integer type variable after an axis name to indirectly specify the coordinate word.

  When the reference unit is set to 0.001 mm with indirect designation using a variable, and the register value is set to 1000, the Machine Controller interprets the coordinate word as 1.000 mm in the same way as for an integer value set in the above example.

◀ EXAMPLE ▶

```
ML00000 = 1000;
MVS  [A1]ML00000;   → 1.000 mm
```



The coordinate word unit depends on the command and motion module settings. Refer to *7.2.2 Reference Unit* for information on the coordinate word unit.

(5) Specific Characters

The meaning and application examples of each specific character are listed below. Refer to the Reference Section indicated in the following table for details of each specific character.

Character	Meaning	Application Examples	Reference Section
F	Interpolation feed speed	MVS [A1]1000 [B1]2000 ***F***3000000; MVS [A1]1000 [B1]2000 ***F***ML00000;	*8.2.2 Linear Interpolation (MVS)*
T	Max. interpolation feed speed	FMX ***T***30000000; FMX ***T***ML00000;	*8.1.7 Maximum Interpolation Feed Speed Setting (FMX)*
	Time Settings	TIM ***T***100; TIM ***T***MW00000; MVT [A1]1000 [B1]2000 ***T***100; MVT [A1]1000 [B1]2000 ***T***ML00000; IAC ***T***100; IAC ***T***ML00000; IDC ***T***100; IDC ***T***ML00000;	*8.4.11 Dwell Time (TIM)* *8.2.9 Set Time Positioning (MVT)* *8.1.9 Interpolation Acceleration Time Change (IAC)* *8.1.10 Interpolation Deceleration Time Change (IDC)*
	Number of turns for circular interpolation	MCW [A1]1000 [B1]2000 U500 V500 ***T***2 F3000000; MCW [A1]1000 [B1]2000 U500 V500 ***T***ML00000 F3000000;	*8.2.3 Clockwise/Counterclockwise Circular Interpolation with Center Position Designation (MCW, MCC)* *8.2.4 Clockwise/Counterclockwise Circular Interpolation with Radius Designation (MCW, MCC)*
R	Radius of circle	MCW [A1]1000 [B1]2000 ***R***500 F3000000; MCW [A1]1000 [B1]2000 ***R***ML00000 F3000000;	*8.2.3 Clockwise/Counterclockwise Circular Interpolation with Center Position Designation (MCW, MCC)* *8.2.4 Clockwise/Counterclockwise Circular Interpolation with Radius Designation (MCW, MCC)*
U	Circle center point coordinate 1 (horizontal axis)	MCW [A1]1000 [B1]2000 ***U***500 V500 T2 F3000000; MCW [A1]1000 [B1]2000 ***U***ML00000 V500 T2 F3000000;	*8.2.3 Clockwise/Counterclockwise Circular Interpolation with Center Position Designation (MCW, MCC)* *8.2.4 Clockwise/Counterclockwise Circular Interpolation with Radius Designation (MCW, MCC)*
V	Circle center point coordinate 2 (vertical axis)	MCW [A1]1000 [B1]2000 U500 ***V***500 T2 F3000000; MCW [A1]1000 [B1]2000 U500 ***V***ML00000 T2 F3000000;	*8.2.3 Clockwise/Counterclockwise Circular Interpolation with Center Position Designation (MCW, MCC)* *8.2.4 Clockwise/Counterclockwise Circular Interpolation with Radius Designation (MCW, MCC)*
P	Interpolation feed speed specified by percentage	IFP ***P***50; IFP ***P***ML00000;	*8.1.8 Interpolation Feed Speed Ratio Setting (IFP)*
SS	Skip signal selection	SKP [A1]1000 [B1]2000 F3000000 ***SS***1; SKP [A1]1000 [B1]2000 F3000000 ***SS***2;	*8.2.8 Linear Interpolation with Skip Function (SKP)*
D	Movement amount for external positioning	EXM [A1]1000 ***D***1000; EXM [A1]1000 ***D***ML00000;	*8.2.10 External Positioning (EXM)*
N	Number of shifts	SFR MB001000 ***N***5 W10; SFR MB001000 ***N***MW00000 W10;	*8.8.1 Bit Right Shift (SFR)* *8.8.2 Bit Left Shift (SFL)*
W	Bit width	BLK MW00100 DW00100 ***W***10; BLK MW00100 DW00100 ***W***MW00000;	*8.8.1 Bit Right Shift (SFR)* *8.8.2 Bit Left Shift (SFL)* *8.8.3 Block Move (BLK)* *8.8.4 Clear (CLR)*

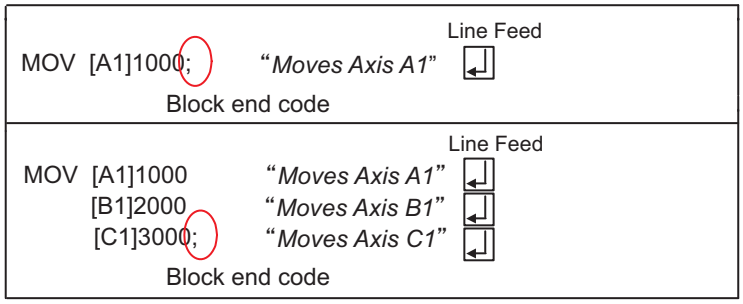
Character	Meaning	Application Examples	Reference Section
MPS	Motion subprogram number	MSEE ***MPS***002;	*8.4.5 Motion Subprogram Call (MSEE)*
SPS	Sequence subprogram number	SSEE ***SPS***002;	*8.4.6 Sequence Subprogram Call (SSEE)*

(6) End of Block

Insert a semi-colon at the end of block. There is no limitation on the number of lines in a block. Code the end of a block to specify its end.

Always insert Line Feed after the end of a block.

◀ EXAMPLE ▶ End of Block Coding Example

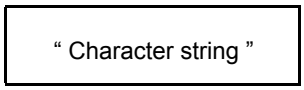


(7) Comments

Either of the following two formats can be used.

- Code a Character String Enclosed in Double Quotation Marks

A character string enclosed in double quotation marks is interpret as a comment.

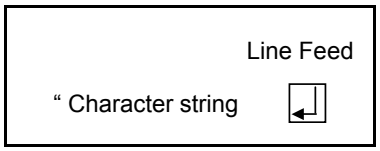


◀ EXAMPLE ▶ Comment Coding Example 1

```
ZRN [A1]0 [B1]0 [C1]0;                  " Zero point return of all axes "
MVS [A1]100.0 [B1]200.0 [C1]300.0;      " 3-axis linear interpolation "
```

- Code a Character String after One Double Quotation Mark

The characters from the double quotation mark to a Line Feed (until the Enter key is pressed) are interpret as a comment.



◀ EXAMPLE ▶ Comment Coding Example 2

```
" Zero point return of all axes
ZRN [A1]0 [B1]0 [C1]0;

" 3-axis linear interpolation
MVS [A1]100.0 [B1]200.0 [C1]300.0;
```

7.1.3 Using Constants and Variables

(1) Constants

The constants that can be used in motion programs are listed below.

Classification	Range	Coding Examples
Decimal Integers	-2147483648 to 2147483647	823, -2493, 123k, 123K
Hexadecimal Integers	00000000H to FFFFFFFFH	FFFABCDEH, 2345H, FH
Real Numbers	-214748.3648 to 214748.3647 Change according to the setting of the number of digits below the decimal point	763.0, 824.2, 234.56, -321.12345

INFO

- The – (minus) sign cannot be omitted, but the + (plus) sign can.

  [A1]+123 ⇒ [A1]123
  [A1]−123 ⇒ [A1]-123

- A decimal integer value is multiplied by 1000 by adding K to the value. For a value such as position reference, adding a K in place of three zeros makes it easy to read.

  [A1]123k ⇒ [A1]123000
  [A1]123K ⇒ [A1]123000

(2) Variables

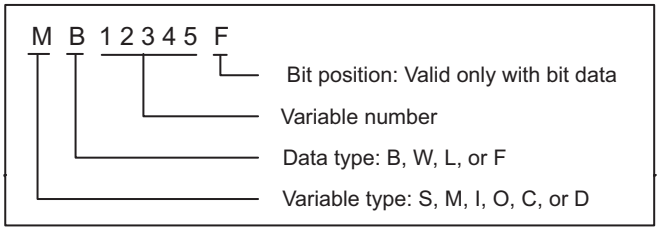
The variables that can be used in motion programs are listed below.

Classification	Variable Type	Data Type			
		Bit	Word	Long	Floating point
Global Variables	S register	SB	SW	SL	SF
	M register	MB	MW	ML	MF
	I register	IB	IW	IL	IF
	O register	OB	OW	OL	OF
	C register	CB	CW	CL	CF
Local Variables	D register	DB	DW	DL	DF

Refer to *Chapter 6 Variables (Registers)* for details on variables.

◀ EXAMPLE ▶

A variable coding example is shown below.





0 (zero) cannot be omitted in all constants and variables.

■ Examples Where 0 (zero) Can be Omitted

```
[A1]00123        =>  [A1]123
[A1]MW00010      =>  [A1]MW10
[A1]100.000      =>  [A1]100.
```

■ Examples Where 0 (zero) Cannot be Omitted

```
MPM001;(Program number coded at the head of program)
MSEE MPS002;
```

7.2 Motion Module Parameters

Motion control using motion programs is determined according to the settings of the motion module parameters. Set the motion module parameters referring to the connected machine before running motion programs.
This chapter describes the minimum settings of motion module parameters required for motion control using motion programs.

7.2.1 Axis Type Selection

There are two types of position control: Finite length position control for return and other operations that are performed only within a specified range, and infinite length position control, which is used for moving in one direction only. Infinite length position control can reset the position to 0 after one rotation, e.g, belt conveyors, or move in one direction only, without resetting position after one rotation. The axis type selection sets which of these types of position control is to be used.
The details of the Axis Type Selection are listed in the following table.

Parameter Type	Parameter No. (Register No.)	Name	Description	Default Value
Motion Fixed Parameters	No. 1, bit 0	Function Selection Flag 1, Axis Selection	Specify the position control method for the controlled axis. 0: Finite Length Axis Set a finite length axis if control is performed within a limited length or for an axis that uses infinite length control in one moving direction only without resetting the position every rotation. 1: Infinite Length Axis Set an infinite length axis for an axis that uses infinite length control while resetting the position every rotation.	0
	No. 10	Infinite Length Axis Reset Position (POSMAX)	Set the reset position of the position data when an infinite length axis has been set for the axis type using the reference unit.	360000

7.2.2 Reference Unit

The unit of position reference that is input from a motion program is called a reference unit, and is a pulse, mm, deg, inch, or μm. The reference unit is specified in Reference Unit Selection (motion fixed parameter 4).

The minimum reference unit that can be specified is determined by the setting of Number of Digits below Decimal Point (motion fixed parameter 5).

Motion Fixed Parameter 5: Number of Digits below Decimal Point	Motion Fixed Parameter 4: Reference Unit Selection				
	0: pulse	1: mm	2: deg	3: inch	4: μm
0: 0 digits	1 pulse*	1 mm	1 deg	1 inch	1μm
1: 1 digits		0.1 mm	0.1 deg	0.1 inch	0.1 μm
2: 2 digits		0.01 mm	0.01 deg	0.01 inch	0.01μm
3: 3 digits		0.001 mm	0.001 deg	0.001 inch	0.001 μm
4: 4 digits		0.0001 mm	0.0001 deg	0.0001 inch	0.0001μm
5: 5 digits		0.00001 mm	0.00001 deg	0.0001 inch	0.00001μm

Minimum reference unit

* When "pulse" is selected, the motion fixed parameter 5 (number of digits below decimal point) is invalid.



The range of reference positions for an axis move command are as follows:

Motion Fixed Parameter 5: Number of Digits below Decimal Point	Motion Fixed Parameter 4: Reference Unit Selection	
	0: pulse	1: mm, 2: deg, 3: inch, 4μm
0: 0 digit	-2147483648 to 2147483647	-2147483648 to 2147483647
1: 1 digit	-2147483648 to 2147483647	-214748364.8 to 214748364.7
2: 2 digits	-2147483648 to 2147483647	-21474836.48 to 21474836.47
3: 3 digits	-2147483648 to 2147483647	-2147483.648 to 2147483.647
4: 4 digits	-2147483648 to 2147483647	-214748.3648 to 214748.3647
5: 5 digits	-2147483648 to 2147483647	-21474.83648 to 21474.83647

7.2.3 Electronic Gear

In contrast to the reference unit input to the Machine Controller, the moving unit in the mechanical system is called the “output unit.” The electronic gear converts position or speed units from reference units to output units for the mechanical system without going through an actual mechanism, such as a gear.

When the axis of the motor has rotated m times and the mechanical configuration allows the axis at the load to rotate n times, this electronic gear function can be used to make the reference unit equal to the output unit.

The electronic gear function is enabled when the following settings are made:

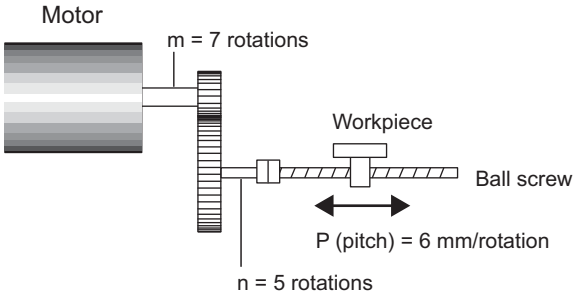
- Fixed Parameter 6: Travel Distance per Machine Rotation
- Fixed Parameter 8: Servo Motor Gear Ratio
- Fixed Parameter 9: Machine Gear Ratio

Note: The electronic gear is disabled when the pulse is specified for the Reference Unit Selection.

The following setting example uses ball screw and rotating table workpieces.

(1) Parameter Setting Example Using Ball Screw

- Machine specifications: Ball screw axis rotates 5 times for every 7 rotations of the motor axis (Refer to the following figure.)
- Reference unit: 0.001 mm



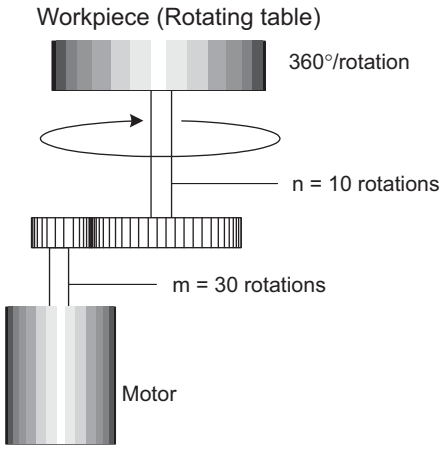
To move the workpiece 0.001 mm for 1 reference unit input under the conditions outlined above, i.e., 1 reference unit = 1 output unit, set the fixed parameters 6, 8, and 9 as follows:

- Fixed Parameter 6: Travel Distance per Machine Rotation = 6 mm/0.001 mm = 6000 (reference units)
- Fixed Parameter 8: Servo Motor Gear Ratio = m = 7
- Fixed Parameter 9: Machine Gear Ratio = n = 5

  Note: Set the SERVOPACK gear ratio to 1:1.

(2) Parameter Setting Example Using Rotating Table

- Machine specifications: Rotating table axis rotates 10 times for every 30 rotations of the motor axis (Refer to the following figure.)
- Reference unit: 0.1°



To rotate the table 0.1° for 1 reference unit input under the conditions outlined above, i.e., 1 reference unit = 1 output unit, set the fixed parameters 6, 8, and 9 as follows:

- Fixed Parameter 6: Travel Distance per Machine Rotation = 360×/0.1× = 3600 (reference units)
- Fixed Parameter 8: Servo Motor Gear Ratio = m = 30
- Fixed Parameter 9: Machine Gear Ratio = n = 10

Note: 1. The gear ratio for fixed parameters 8 and 9 (m/n) may be constant, e.g., m = 3 and n = 1.
2. Set the SERVOPACK gear ratio to 1:1.

7.2.4 Speed Reference

The unit for the speed coded in a motion program can be selected: Reference unit/s, 10^n reference unit/min., 0.01% of rated speed, or 0.0001% of rated speed. Use bits 0 to 3 (Speed unit selection) of the motion setting parameter OW□□03 to select the unit.

Motion Setting Parameter	Speed Unit	Coordinate Word Designation Method
OW□□03, bits 0 to 3: Speed unit selection	0: Reference unit/s	Designate a movement amount per second in reference units.
	1: 10^n reference unit/ min.	Designate a movement amount per minute in reference units. When the reference unit is a pulse: The designated value is multiplied by 1000. Example: Reference unit = pulse *VEL [A1]1000;* Speed reference = 1,000,000 [pulses/min.]. When the reference unit is not a pulse: The designated value is handled as: Example) Reference unit = mm *VEL [A1]1000;* Number of digits below decimal point n = 3 Speed reference = 1000×10^3 [0.001 mm/min]
	2: 0.01%	Designates the speed in percentage of the rated speed in units of 0.01%.
	3: 0.0001%	Designates the speed in percentage of the rated speed in units of 0.0001%.

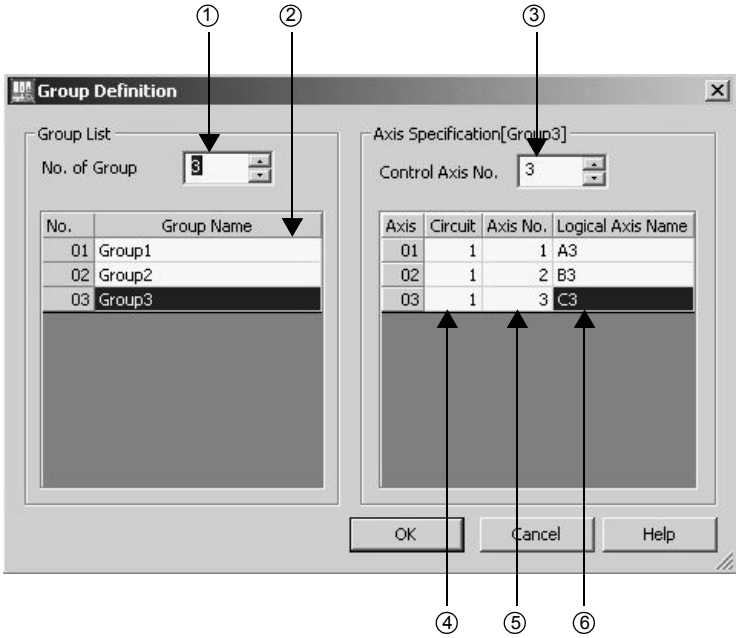
7.2.5 Acceleration/Deceleration Setting

The unit for the acceleration/deceleration rate coded in a motion program can be selected: Reference unit/s^2 or ms. Use bits 4 to 7 (Acceleration/deceleration degree unit selection) of the motion setting parameter OW□□03 to select the unit.

Motion Setting Parameter	Speed Unit	Coordinate Word Designation Method
OW□□03, bits 4 to 7 Acceleration/deceler-ation degree unit selection	0: Reference unit/s^2	Designates the acceleration/deceleration rate per second in units of reference unit/s^2.
	1: ms	Designates the acceleration/deceleration time in units of ms.

7.3 Group Definition

Axes to be grouped together are defined in the **Group Definition** window.
This section describes the components of the **Group Definition** window.



① No. of Group

Set a number for operation as a group.
Set it to 1 for operation as one group.
Set it to the number of groups for operation with multiple groups.

② Group Name

Define a group name.

③ Control Axis No.

Set the number of axes controlled in the group.

④ Circuit

Set a line number for the motion module used.
The line number can be checked in the module configuration definition.

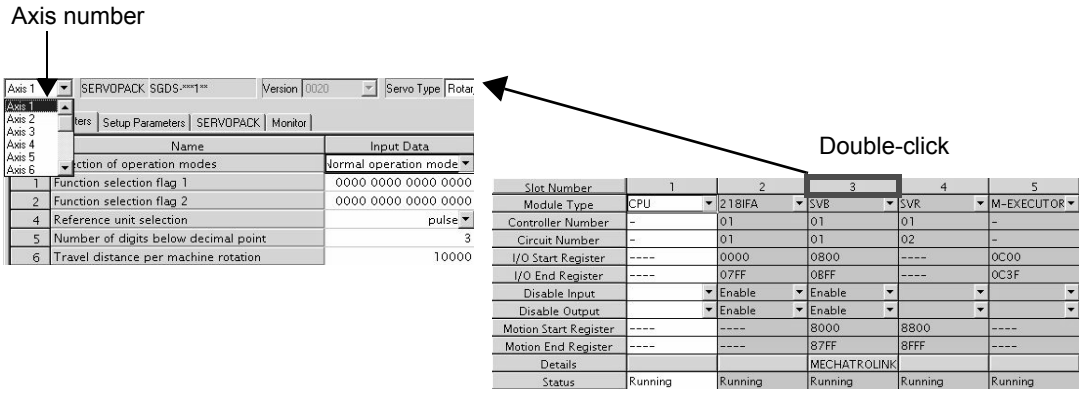
Line number

Slot Number	1	2	3	4	5
Module Type	CPU	218IFA	SVB	SVR	M-EXECUTOR
Controller Number	-	01	01	01	-
Circuit Number	-	01	01	02	-
I/O Start Register	----	0000	0800	----	0C00
I/O End Register	----	07FF	0BFF	----	0C3F
Disable Input		Enable	Enable		
Disable Output		Enable	Enable		
Motion Start Register	----	----	8000	8800	----
Motion End Register	----	----	87FF	8FFF	----
Details			MECHATROLINK		
Status	Running	Running	Running	Running	Running

⑤ Axis No.

Set an axis number for the axis used.

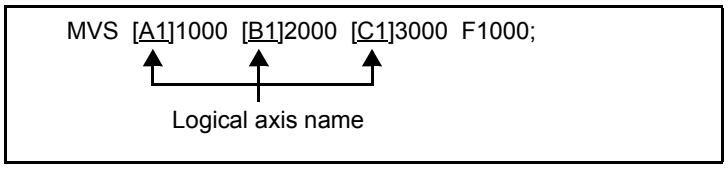
The axis number can be checked in the detailed screen of the used motion module.



⑥ Logical Axis Name

Define a name for the specified axis number.

The name defined here is used when programming a motion program.



7.4 Priority Levels of Operations

A priority level is assigned to each operator used in an operation written in motion language.
For an operation involving three or more items, specify the priority level by using ().
The priority levels of operators are shown below.

High ⟶ Low

Operator	Priority Level			
	1	2	3	4
Parentheses	()			
NOT		!		
AND			&	
OR				|
XOR				^
Arithmetic operation				+ - * /

- Arithmetic Operation Example

◀ EXAMPLE ▶

- Operation example

```
MW00100 = 1 + 2;
```

With this operation, 1 + 2 is calculated, and the result of 3 is stored in MW00100.

- Example of operation involving three items or more

```
MW00100 = 1 +  (2 * 3);
```

With this operation, 2 * 3 is calculated first, and 1 is added to the result of 6. The final result of 7 is then stored in MW00100.
Therefore, MW00100 = 7

INFO ■ Precautions on operations involving three items or more

For example, with the operation below,

```
MW00100 = 1 + 2 * 3;
```

1 + 2 is calculated first according to the priority level shown above, and then the result of 3 is multiplied by 3. The final result of 9 is then stored in MW00100. Therefore, MW00100 = 9.

- Logical Operation Example

◀ EXAMPLE ▶

- Operation example

```
MW00100 = 0001H | 0002H;
```

With this operation, OR operation of 0001H and 0002H are executed, and the result is stored in MW00100.

- Example of operation involving three or more items

```
MW00100 = (1111H | 2222H) & 00FFH;
```

With this operation, OR operation of 1111H and 2222H are executed first, and the AND operation of the OR result and 00FFH is calculated. The AND result is then stored in MW00100.
Therefore, MW00100 = 0033H

INFO

■ Precautions on operations involving three or more items

For example, with the following operation,

```
MW00100 = 1111H | 2222H & 00FFH;
```

With this operation, the AND operation of 2222H and 00FFH is executed first. Then, the OR operation of the AND result and 1111H is executed. The OR result is then stored in MW00100.
Therefore, MW00100 = 1133H

7.5 Commands and Execution Scans

7.5.1 Command Types

There are three motion language command types. The number of scans required to execute a command will differ depending on the command type. The following table shows the number of scans required to execute each type of command.

Command Type	Command	Number of Scans Required to Execute a Command
S type	Operation commands	One scan
M type	Axis move commands	More than one scan
T type	Timer related commands	

The details of each command type follow:

- S Type Commands

  S type commands, including operation commands, are executed in one scan.
  A program in which S type commands are continuously coded is executed within one scan.

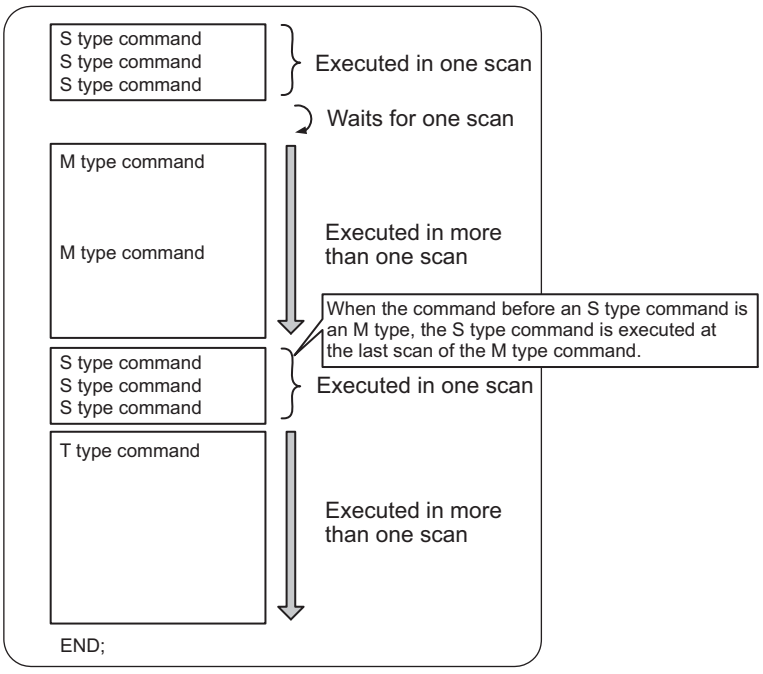
- M Type Commands

  M type commands, including axis move commands, are executed in more than one scan.
  One scan is required to switch from the S type command to the M type command.

- T Type Commands

  T type commands, including timer related commands, are executed in more than one scan.

The following diagram shows the number of scans required to execute each command type.



7.5.2 List of Command Types

The following table lists the command types.

Classification	Command	S Type	M Type	T Type	Classification	Command	S Type	M Type	T Type
Axis Setting Commands	ABS	✓			Arithmetic Operation	=	✓		
	INC	✓				+	✓		
	ACC		✓			-	✓		
	DCC		✓			*	✓		
	SCC		✓			/	✓		
	VEL		✓			MOD	✓		
	FMX	✓			Logical Operation	|	✓		
	IFP	✓				&	✓		
	IAC	✓				^	✓		
	IDC	✓				!	✓		
Axis Move Commands	MOV		✓		Data Comparison	==	✓		
	MVS		✓			<>	✓		
	MCW		✓			>	✓		
	MCC		✓			<	✓		
	ZRN		✓			>=	✓		
	SKP		✓			<=	✓		
	MVT		✓		Data Operations	SFR	✓		
	EXM		✓			SFL	✓		
Control Commands	POS	✓				BLK	✓		
	MVM	✓				CLR	✓		
	PLN	✓				ASCII	✓		
	PLD	✓			Basic Functions	SIN	✓		
	PFN		✓			COS	✓		
	INP		✓			TAN	✓		
Program Control Commands	IF ELSE IEND	✓				ASN	✓		
	WHILE WEND	✓				ACS	✓		
	PFORK JOINTO PJOINT	✓				ATN	✓		
	SFORK JOINTO SJOINT	✓				SQT	✓		
	MSEE	✓				BIN	✓		
	SSEE	✓				BCD	✓		
	UFC		✓			S{ }	✓		
	FUNC	✓				R{ }	✓		
	END	✓				PON	✓		
	RET	✓				NON	✓		
	TIM			✓		TON	✓		
	IOW			✓		TOF	✓		
	EOX			✓	C Language Control Commands	CTSK	✓		
	SNGD/ SNGE	✓				CFUNC	✓		

7.6 Sequence Program Format

The format of a sequence program is the same as that for a motion program.
However, the motion language commands that can be used in a sequence program are limited. For the commands that can be used in sequence programs, refer to *Appendix A Motion Language Commands*.

8

Command Reference

This chapter describes the motion language commands.

8.1 Axis Setting Commands ---------- 8-3
8.1.1 Absolute Mode (ABS) ---------- 8-3
8.1.2 Incremental Mode (INC) ---------- 8-7
8.1.3 Acceleration Time Change (ACC) ---------- 8-11
8.1.4 Deceleration Time Change (DCC) ---------- 8-17
8.1.5 S-curve Time Constant Change (SCC) ---------- 8-23
8.1.6 Set Velocity (VEL) ---------- 8-29
8.1.7 Maximum Interpolation Feed Speed Setting (FMX) ---------- 8-35
8.1.8 Interpolation Feed Speed Ratio Setting (IFP) ---------- 8-37
8.1.9 Interpolation Acceleration Time Change (IAC) ---------- 8-40
8.1.10 Interpolation Deceleration Time Change (IDC) ---------- 8-43
8.2 Axis Move Commands ---------- 8-46
8.2.1 Positioning (MOV) ---------- 8-46
8.2.2 Linear Interpolation (MVS) ---------- 8-50
8.2.3 Clockwise/Counterclockwise Circular Interpolation with Center Position Designation (MCW, MCC) ---------- 8-55
8.2.4 Clockwise/Counterclockwise Circular Interpolation with Radius Designation (MCW, MCC) ---------- 8-61
8.2.5 Clockwise/Counterclockwise Helical Interpolation with Center Position Designation (MCW, MCC) ---------- 8-65
8.2.6 Clockwise/Counterclockwise Helical Interpolation with Radius Designation (MCW, MCC) ---------- 8-68
8.2.7 Zero Point Return (ZRN) ---------- 8-70
8.2.8 Linear Interpolation with Skip Function (SKP) ---------- 8-72
8.2.9 Set Time Positioning (MVT) ---------- 8-74
8.2.10 External Positioning (EXM) ---------- 8-76
8.3 Axis Control Commands ---------- 8-78
8.3.1 Current Position Set (POS) ---------- 8-78
8.3.2 Move On Machine Coordinates (MVM) ---------- 8-80
8.3.3 Program Current Position Update (PLD) ---------- 8-81
8.3.4 In-Position Check (PFN) ---------- 8-82
8.3.5 Set In-Position Range (INP) ---------- 8-84
8.3.6 Coordinate Plane Setting (PLN) ---------- 8-86

8.4 Program Control Commands ---------- 8-87
8.4.1 Branching Commands (IF ELSE IEND) ---------- 8-87
8.4.2 Repeat (WHILE WEND) ---------- 8-89
8.4.3 Parallel Execution (PFORK, JOINTO, PJOINT) ---------- 8-92
8.4.4 Selective Execution (SFORK, JOINTO, SJOINT) ---------- 8-95
8.4.5 Motion Subprogram Call (MSEE) ---------- 8-99
8.4.6 Sequence Subprogram Call (SSEE) ---------- 8-100
8.4.7 User Function Call From Motion Program (UFC) ---------- 8-101
8.4.8 User Function Call from Sequence Program (FUNC) ---------- 8-109
8.4.9 Program End (END) ---------- 8-110
8.4.10 Subprogram End (RET) ---------- 8-111
8.4.11 Dwell Time (TIM) ---------- 8-112
8.4.12 I/O Variable Wait (IOW) ---------- 8-113
8.4.13 One Scan Wait (EOX) ---------- 8-115
8.4.14 Single-block Signal Disabled (SNGD)/Single-block Signal Enabled (SNGE) ---------- 8-116
8.5 Arithmetic Operations ---------- 8-117
8.5.1 Substitute (=) ---------- 8-117
8.5.2 Add (+) ---------- 8-118
8.5.3 Subtract (-) ---------- 8-119
8.5.4 Multiply (*) ---------- 8-120
8.5.5 Divide (/) ---------- 8-121
8.5.6 Remainder (MOD) ---------- 8-122
8.6 Logic Operation ---------- 8-123
8.6.1 OR (|) ---------- 8-123
8.6.2 AND (&) ---------- 8-125
8.6.3 XOR (^) ---------- 8-126
8.6.4 NOT (!) ---------- 8-127
8.7 Data Comparisons ---------- 8-128
8.7.1 Data Comparison Commands (==, <>, >, <, >=, <=) ---------- 8-128
8.8 Data Operations ---------- 8-130
8.8.1 Bit Right Shift (SFR) ---------- 8-130
8.8.2 Bit Left Shift (SFL) ---------- 8-131
8.8.3 Block Move (BLK) ---------- 8-132
8.8.4 Clear (CLR) ---------- 8-133
8.8.5 ASCII Conversion 1 (ASCII) ---------- 8-134
8.9 Basic Functions ---------- 8-136
8.9.1 Sine (SIN) ---------- 8-136
8.9.2 Cosine (COS) ---------- 8-138
8.9.3 Tangent (TAN) ---------- 8-139
8.9.4 Arc Sine (ASN) ---------- 8-140
8.9.5 Arc Cosine (ACS) ---------- 8-141
8.9.6 Arc Tangent (ATN) ---------- 8-142
8.9.7 Square Root (SQT) ---------- 8-143
8.9.8 BCD to Binary (BIN) ---------- 8-145
8.9.9 Binary to BCD (BCD) ---------- 8-146
8.9.10 Set Bit (S{ }) ---------- 8-147
8.9.11 Reset Bit (R{ }) ---------- 8-148
8.9.12 Rising Pulse (PON) ---------- 8-149
8.9.13 Falling Pulse (NON) ---------- 8-151
8.9.14 ON-Delay Timer (TON): Counting unit: 0.01 second ---------- 8-154
8.9.15 OFF-Delay Timer (TOF):Counting unit: 0.01 second ---------- 8-156
8.10 C-Language Control Commands ---------- 8-158
8.10.1 C-Language Task Control (CTSK) ---------- 8-158
8.10.2 C-Language Function Call (CFUNC) ---------- 8-160

8.1 Axis Setting Commands

This section describes the axis setting commands.

8.1.1 Absolute Mode (ABS)

Motion Programs	Sequence Programs
Available	Not Available

⚠ CAUTION

- The movement of a coordinate word designated in ABS mode is entirely different from that of the same coordinate word designated in INC mode. Before starting operations, be sure to check that the ABS or INC command is specified correctly.
  Failure to carry out this check may result in damage to equipment, serious personal injury, or even death.

(1) Overview

The Absolute Mode (ABS) command causes the coordinate words that control axis movement to be treated as a target position.

Once ABS mode has been executed, it remains in effect until Incremental Mode (INC) is next executed. ABS mode is the default mode when the program operation is started.

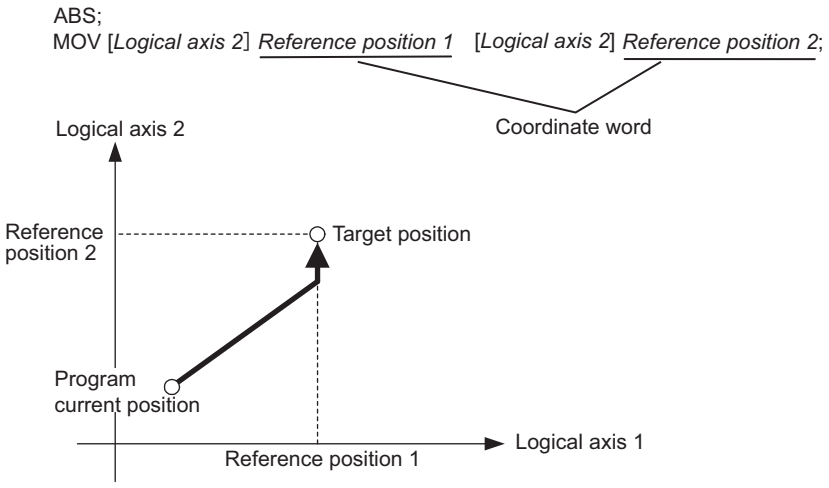


Fig. 8.1 Movement Mode: Absolute Mode (ABS)

In this manual, the coordinate word that follows the logical axis name for the axis move command is expressed as "reference position" or "position reference value."

TERMS

■ Program Current Position

The position on the work coordinate system when an axis is started moving by execution of an axis move command. And, the work coordinate system is the coordinate system used in motion programs.
For information on the work coordinate system, refer to *8.3.1 Current Position Set (POS)*.

(2) Format

- To code an individual ABS command
  ABS;

- To code an ABS command in the same block with an axis move command
  ABS MOV [*Logical axis name 1*] — [*Logical axis name 2*] — ;

(3) Programming Example

An ABS command programming example is given below.

◀ EXAMPLE ▶

```
ABS;                          "Absolute mode
MOV [A1]10000 [B1]40000;      "Positioning
MOV [A1]50000 [B1]20000;      "Positioning
END;
```

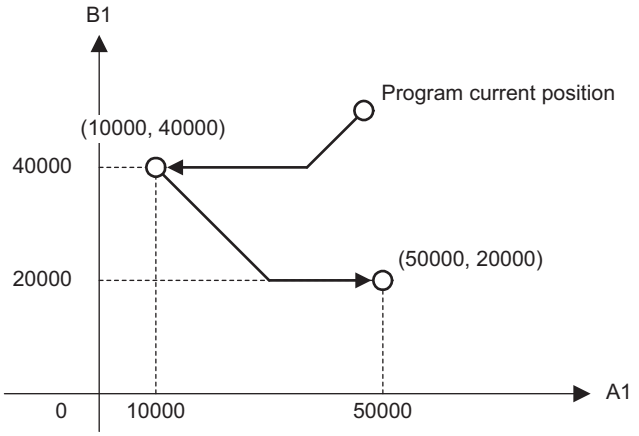


Fig. 8.2 ABS Command Programming Example

(4) Supplemental Information on ABS Command

(a) Related Motion Parameters

Motion setting parameters and ABS commands have no relation.
The movement mode (ABS mode/INC mode) for axis move commands is control data exclusively reserved for motion programs, and no motion setting parameter can be used to specify these modes.

INFO

- Note that the movement mode (ABS mode/INC mode) for axis move commands is totally different from the position reference type specified by the motion setting parameter OW□□09, bit 5.
- When executing a motion program, set bit 5 (position reference type) of motion setting parameter OW□□09 to 0 (incremental addition mode) regardless of the movement mode setting.

(b) Finite-length Axis and Infinite-length Axis

The position reference value of a coordinate word for a finite-length axis must be handled differently from one for an infinite-length axis.
The following table shows how to designate the position reference values for a finite-length axis and infinite-length axis.

Axis Type	Movement Mode for Axis Move Command	Position Reference Value Designation Method
Finite-length Axis	ABS Mode	Designate the target position for the position reference value.
	INC Mode	Designate the relative movement amount for the position reference value.
Infinite-length Axis	ABS Mode	Designate the target position in the range between 0 to POSMAX for the position reference value. The position reference value indicates the movement direction: Positive direction with a positive value, and negative direction with a negative value.
	INC Mode	Designate the relative movement amount for the position reference value.

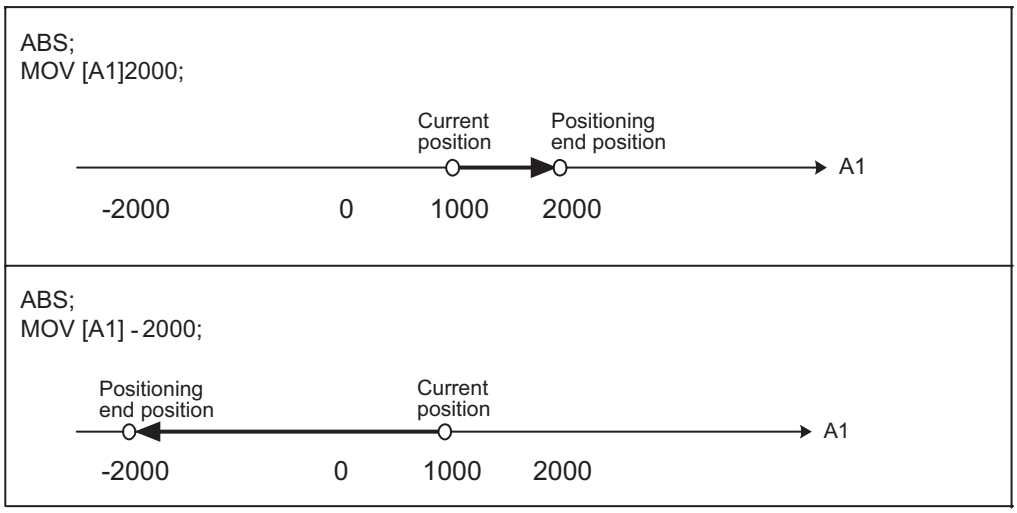
INFO

- Use bit 0 (Axis selection) of motion fixed parameter 1 (Function selection flag 1) to select a finite-length axis or infinite-length axis.
  Select a finite-length axis or infinite-length axis according to the machine configuration. For details on how to set motion parameters, refer to the manual for the motion module to be used.
- Use the motion fixed parameter 10 (Infinite length axis reset position) to set POSMAX.

The motions of the finite-length axis and infinite-length axis in ABS mode are shown below.
For the motions in INC mode, refer to *8.1.2 Incremental Mode (INC)*.

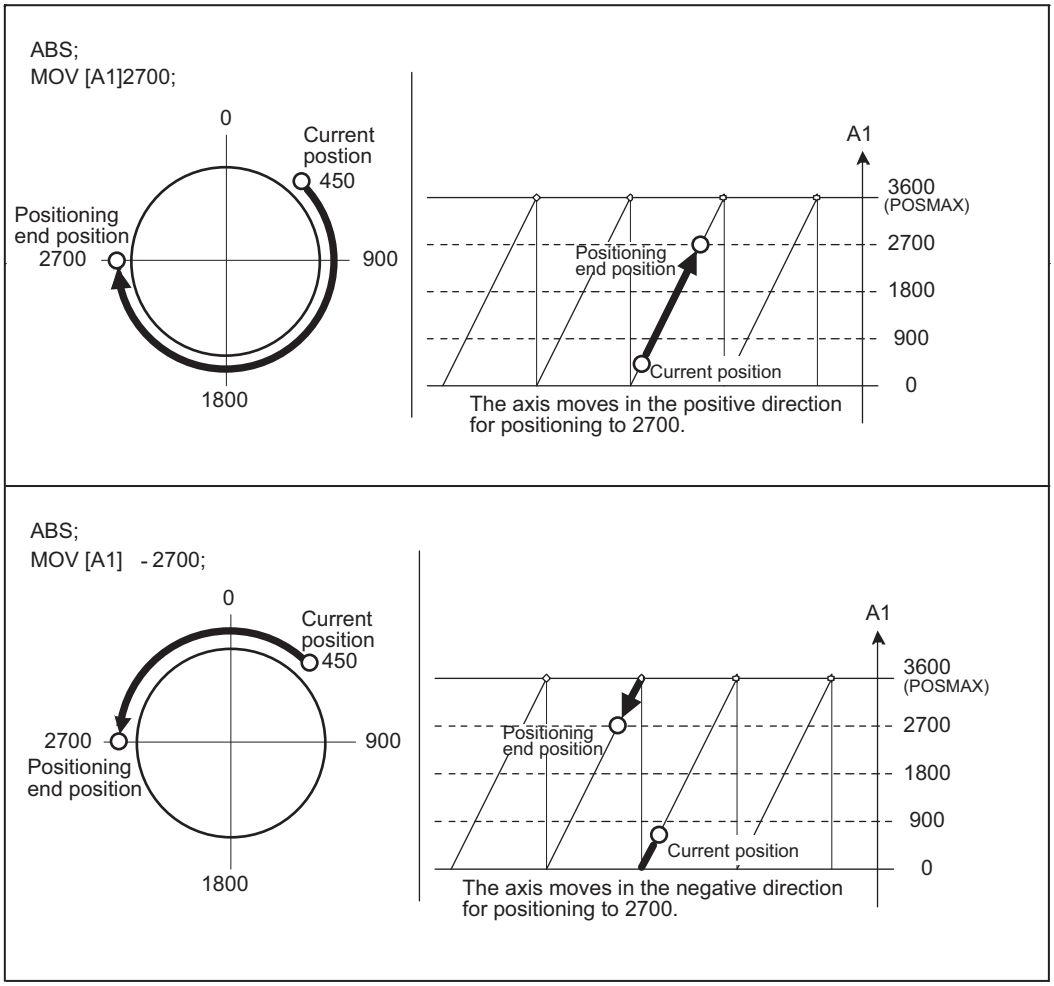
• When Selecting ABS Mode for a Finite-length Axis

Specify the target position for the position reference value.
The following diagrams illustrate the axis motions when specifying the position reference values 2000 and -2000 from the current position (1000).



- When Selecting ABS Mode for an Infinite-length Axis

Specify the target position in the range between 0 to POSMAX for the position reference value.
The position reference value indicates the movement direction: Positive direction with a positive value, and negative direction with a negative value.
The following diagrams illustrate the axis motions when specifying the position reference values 2700 and -2700 from the current position 450 for the infinite-length axis, with POSMAX set to 3600.



INFO

- When the position reference value +0 is specified for an infinite-length axis in ABS mode, the axis moves in the negative direction.
  Specify the POSMAX value for the position reference value to move the axis in the positive direction.
- If the target position (the absolute value of the position reference value) exceeds the POSMAX value for an infinite-length axis in ABS mode, an alarm will occur in the motion program.

8.1.2 Incremental Mode (INC)

Motion Programs	Sequence Programs
Available	Not Available

⚠ CAUTION

- The movement of a coordinate word designated in ABS mode is entirely different from that of the same coordinate word designated in INC mode. Before starting operations, be sure to check that the ABS or INC command is specified correctly.
  Failure to carry out this check may result in damage to equipment, serious personal injury, or even death.

(1) Overview

The Incremental Mode (INC) command causes the coordinate words that control axis movement to be treated as a relative movement amount.
Once INC mode has been executed, it remains in effect until the next time the Absolute Mode (ABS) is executed. ABS mode is the default mode when the program operation starts.

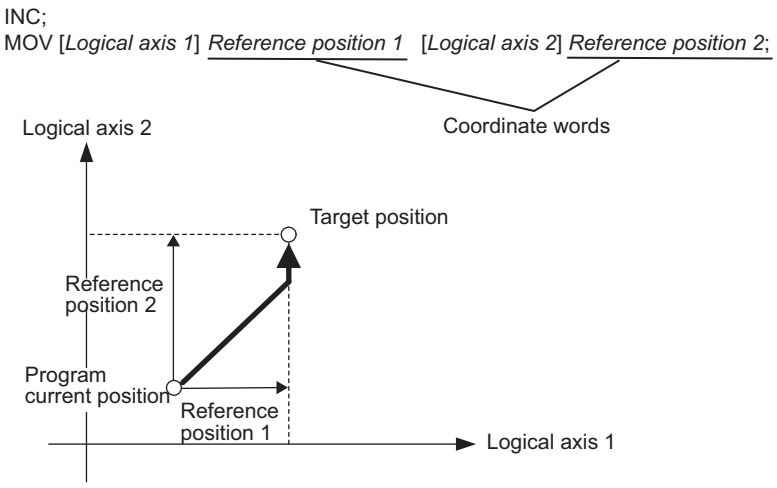


Fig. 8.3 Movement Mode of Incremental Mode (INC)

In this manual, the coordinate word that follows the logical axis name for the axis move command is expressed as "reference position" or "position reference value."

TERMS

■ Program Current Position

The position on the work coordinate system when an axis is started moving by execution of an axis move command. And, the work coordinate system is the coordinate system used in motion programs.
For information on the work coordinate system, refer to *8.3.1 Current Position Set (POS)*.

(2) Format

- To code an individual INC command
  INC;

- To code an INC command in the same block with an axis move command
  INC MOV [*Logical axis name 1*] — | [*Logical axis name 2*] — ;

(3) Programming Example

An INC command programming example is given below.

◀ EXAMPLE ▶

```
INC;                          " Incremental mode
MOV [A1]20000 [B1]30000;      " Positioning
MOV [A1]20000 [B1]10000;      " Positioning
END;
```

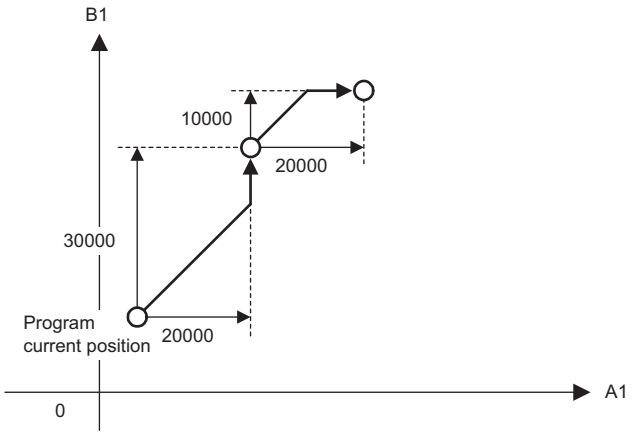


Fig. 8.4 INC Command Programming Example

(4) Supplemental Information on INC Command

(a) Related Motion Parameters

Motion setting parameters and INC commands have no relation.

The movement mode (ABS mode/INC mode) for axis move commands is control data exclusively reserved for motion programs, and no motion setting parameter can be used to specify these modes.

(b) Finite-length Axis and Infinite-length Axis

The position reference value of a coordinate word for a finite-length axis must be handled differently from one for an infinite-length axis.
The following table shows how to designate the position reference values for a finite-length axis and infinite-length axis.

Axis Type	Movement Mode for Axis Move Command	Position Reference Value Designation Method
Finite-length Axis	ABS Mode	Designate the target position for the position reference value.
	INC Mode	Designate the relative movement amount for the position reference value.
Infinite-length Axis	ABS Mode	Designate the target position in the range between 0 to POSMAX for the position reference value. The position reference value indicates the movement direction: Positive direction with a positive value, and negative direction with a negative value.
	INC Mode	Designate the relative movement amount for the position reference value.

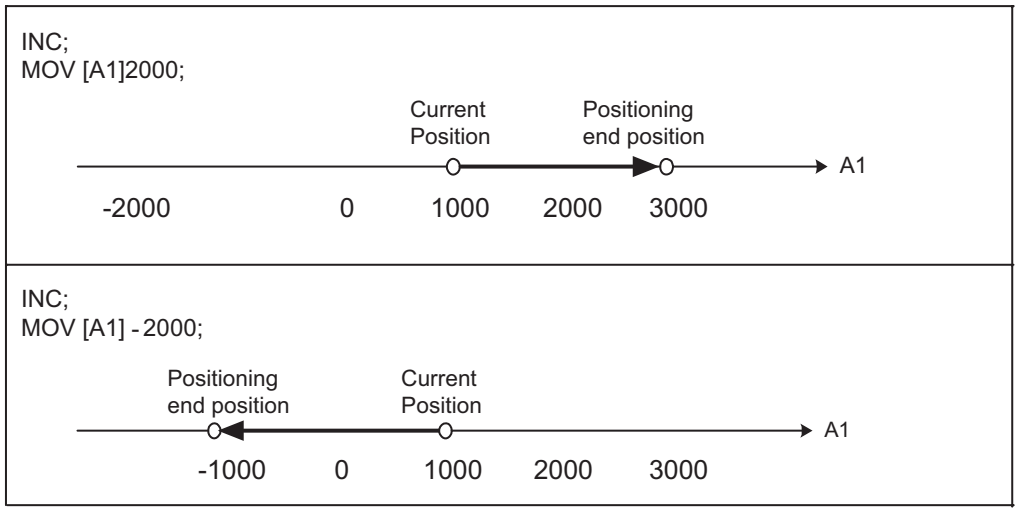
INFO

- Use bit 0 (Axis selection) of motion fixed parameter 1 (Function selection flag 1) to select a finite-length axis or infinite-length axis.
  Select a finite-length axis or infinite-length axis according to the machine configuration. For details on how to set motion parameters, refer to the manual for the motion module to be used.
- Use the motion fixed parameter 10 (Infinite length axis reset position) to set POSMAX.

The motions of the finite-length axis and infinite-length axis in INC mode are shown below.
For the motions in ABS mode, refer to *8.1.1 Absolute Mode (ABS)*.

• When Selecting INC Mode for a Finite-length Axis

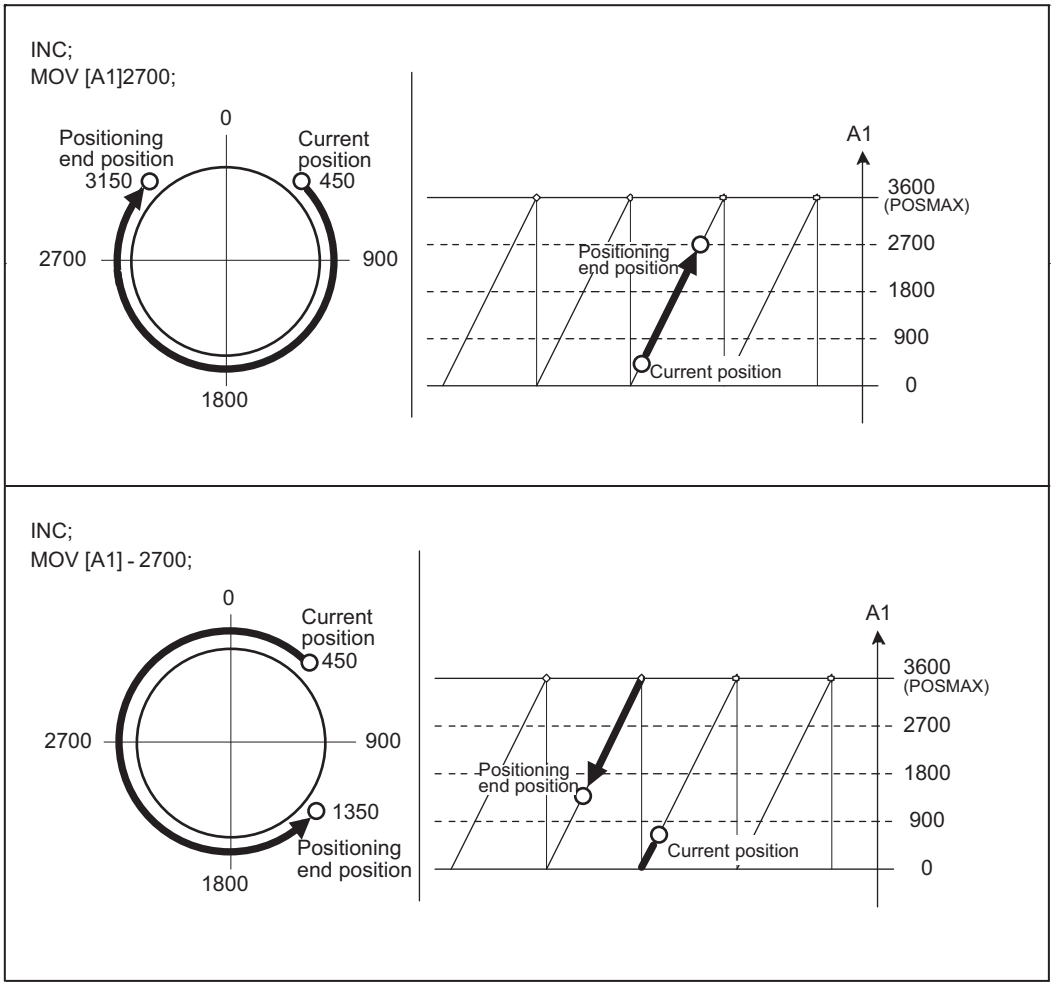
Specify the relative movement amount for the position reference value.
The following diagrams illustrate the axis motions when specifying the position reference values 2000 and -2000 from the current position (1000).



- When Selecting INC Mode for Infinite-length Axis

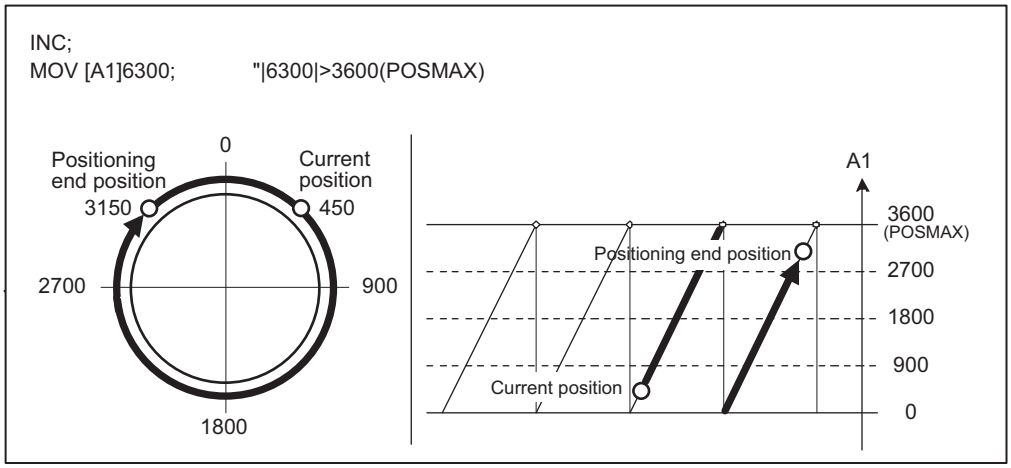
Specify the relative movement amount for the position reference value.

The following diagrams illustrate the axis motions when specifying the position reference values 2700 and -2700 from the current position 450 for the infinite-length axis, with POSMAX set to 3600.



INFO

- If the absolute value of the position reference value (coordinate word) exceeds the POSMAX value, the position reference value (coordinate word) is used for the relative movement amount to move the axis in INC mode.



8.1.3 Acceleration Time Change (ACC)

Motion Programs	Sequence Programs
Applicable	Not applicable

(1) Overview

The Acceleration Time Change command (ACC) changes the acceleration time or acceleration rate of the axis for which one of the following axis move commands is executed.

- Positioning (MOV)
- Set Time Positioning (MVT)
- External Positioning (EXM)

The acceleration time for up to 16 axes can be changed at once. The acceleration time of any axis unspecified in the command block is not changed.
The acceleration time changed by the ACC command remains until it is reset by the next ACC command.

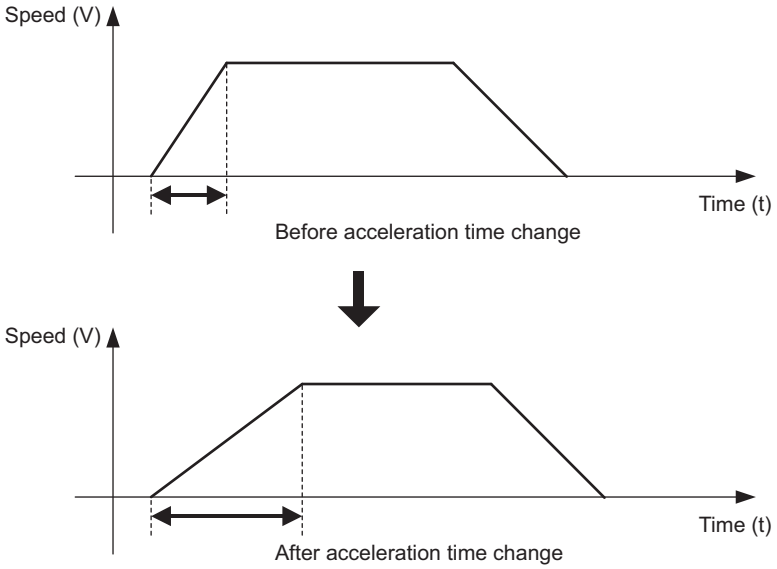


Fig. 8.5 Acceleration Time Change (ACC)

INFO

- The Acceleration Time Change command (ACC) changes the acceleration time for positioning related commands MOV, EXM, and MVT. Use an IAC command to set the acceleration time for interpolation related commands, MVS, MCW, MCC, and SKP.
- All motion modules, except the PO-01 module, support ACC, DCC, and SCC commands.
  Use of these commands with the PO-01 module requires the following combination of MP2000-series Machine Controller and PO-01 module versions.

MP2000 Series Version Number	PO-01 Module Version Number
Ver.2.46 or later	Ver.1.06 or later

(2) Format

ACC [*Logical axis name 1*] *Acceleration time* [*Logical axis name 2*] *Acceleration time* [*Logical axis name 3*] *Acceleration time* . . . ;

Item	Unit	Data That Can Be Used
Acceleration time or acceleration rate	ms or reference unit/s^2 Note: Use bits 4 to 7 of the motion setting parameter OW□□03 (Acceleration/deceleration degree unit selection) to set the unit.	· Directly specified value · Double integer register (Indirect designation)

(3) Setting Items for ACC Command

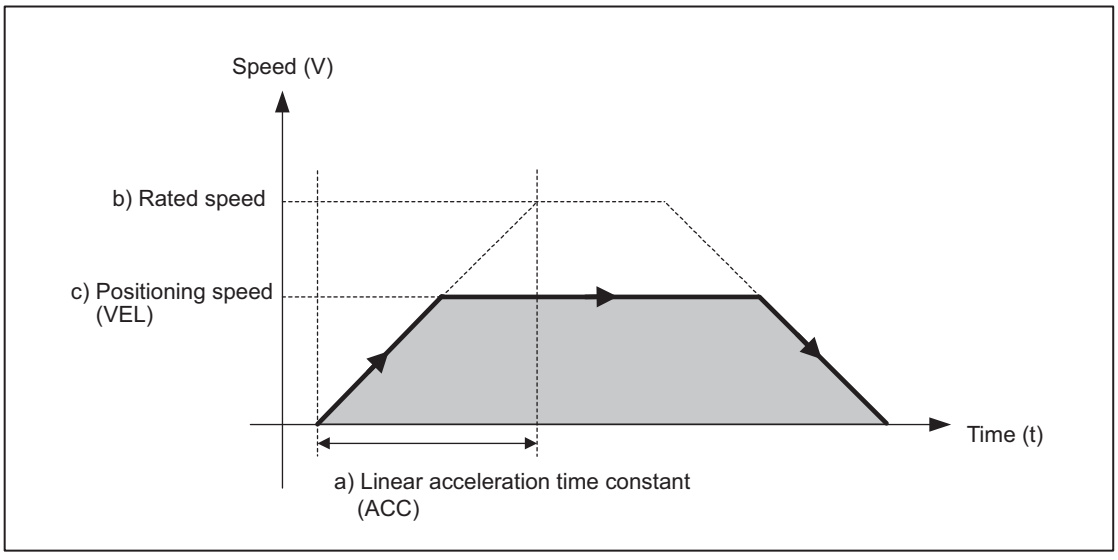
Either acceleration time (ms) or acceleration rate (reference unit/s^2) can be selected for the unit of set value for ACC command.

Motion setting parameter OW□□03, bits 4 to 7 (Acceleration/deceleration degree unit selection)

Parameter Name	Acceleration/Deceleration Unit
Function setting 1, Acceleration/deceleration degree unit selection	0: Reference unit/s^2 1: ms (default)

- When bits 4 to 7 (Acceleration/deceleration degree unit selection) of OW□□03 are set to 1 (ms)

Motion Image



a) Linear acceleration time constant

Set the linear acceleration time constant (the time required to reach the rated speed from 0) for the ACC command. The reference range is as follows.

1 to 32767 (ms)

b) Rated speed

Set the rated speed of each axis using the motion fixed parameter 34 (Rated motor speed).
For details, refer to the manual for the motion module to be used.

c) Positioning speed

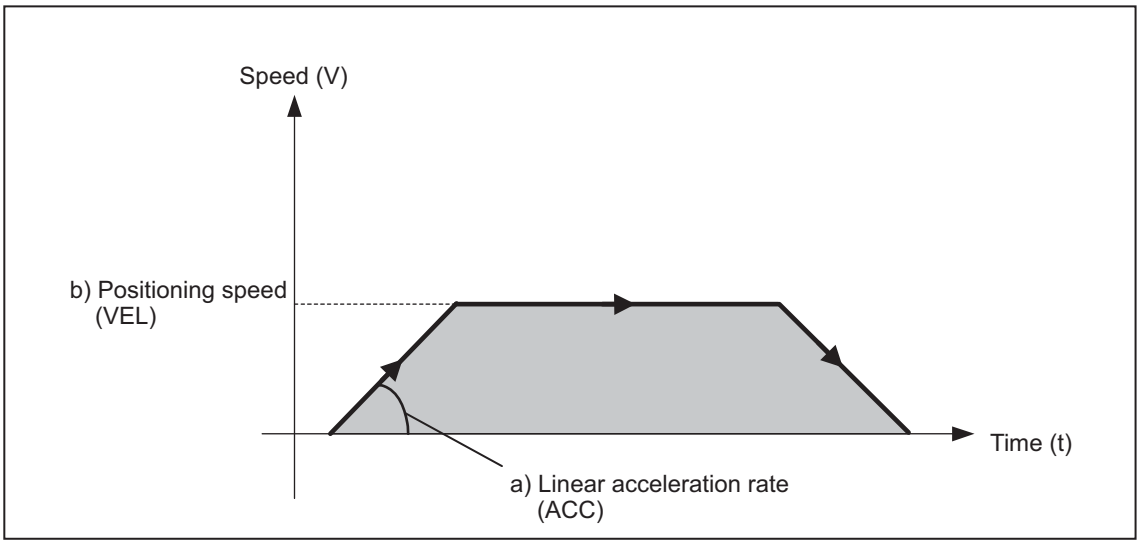
The speed for positioning related commands MOV, MVT, or EXM.
Set the positioning speed of each axis using the Set Velocity command (VEL).

INFO

For the Set Time Positioning command (MVT), the positioning speed is not the reference value of the VEL command. The Set Time Positioning command (MVT) changes the positioning speed according to the set positioning time and the amount of movement.

- When bits 4 to 7 (Acceleration/deceleration degree unit selection) of OW□□03 are set to 0 (reference unit/s^2)

Motion Image



a) Linear acceleration rate

Set the value of the linear acceleration rate for the ACC command.
The reference range is as follows.

1 to 2^{31}-1 (reference unit/s^2)

b) Positioning speed

The speed of positioning related commands, MOV, MVT, or EXM.
Set the positioning speed of each axis using the Set Velocity command (VEL).

INFO

For the Set Time Positioning command (MVT), the positioning speed is not the reference value of the VEL command. The Set Time Positioning command (MVT) changes the positioning speed according to the set positioning time and the amount of movement.

(4) Programming Examples

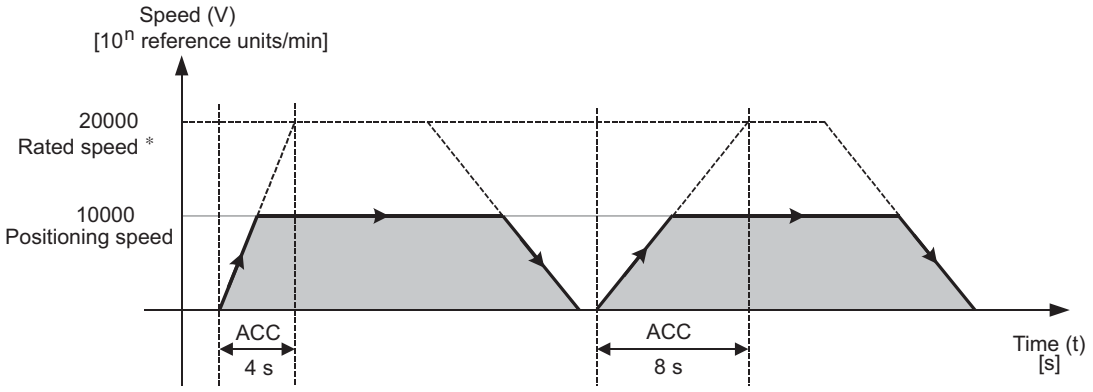
ACC command programming examples are shown below.

◀ EXAMPLE ▶

- When bits 4 to 7 (Acceleration/deceleration degree unit selection) of OW□□03 are set to 1 (ms)

The following example shows when the MOV command is executed to accelerate axis A1 from static to the rated speed in four seconds, and when the MOV command is executed to accelerate axis A1 in eight seconds.

```
INC;                    " Incremental mode
VEL [A1]10000;          " Set Velocity [10**n reference units/min.]
DCC [A1]8000;           " Deceleration time change [ms]
ACC [A1]4000;           " Acceleration time change [ms]
MOV [A1]5000000;        " Positioning
DL00000 = 8000;         " Acceleration time [ms]
ACC [A1]DL00000;        " Acceleration time change [ms]
MOV [A1]5000000;        " Positioning
END;
```



* The unit of rated speed (min^{-1}) must be converted to the same unit used for positioning speed (10^n reference units/min).

Fig. 8.6 Acceleration Time Change (ACC) Command Programming Example 1:
When Acceleration/Deceleration Degree Unit Selection is Set to 1 (ms)

◀ EXAMPLE ▶

- When bits 4 to 7 of OW□□03 (Acceleration/deceleration degree unit selection) are set to 0 (reference unit/s^2)

The following example shows when the MOV command is executed to accelerate the axis A1 at the rate of 60.000 (mm/s^2), and when the MOV command is executed to accelerate axis A1 at the rate of 100.000 (mm/s^2). In this example, one reference unit is set to 0.001 mm.

```
INC;                  " Incremental mode
VEL [A1]18000;        " Set Velocity [10**n reference units/min]
DCC [A1]100000;       " Deceleration time change [reference unit /S*S]
ACC [A1]60000;        " Acceleration rate change [reference unit/S*S]
MOV [A1]5000000;      " Positioning
DL00000 = 100000;     " Acceleration rate [reference unit/S*S]
ACC [A1] DL00000;     " Acceleration rate change [reference unit/S*S]
MOV [A1]5000000;      " Positioning
END;                  END;
```

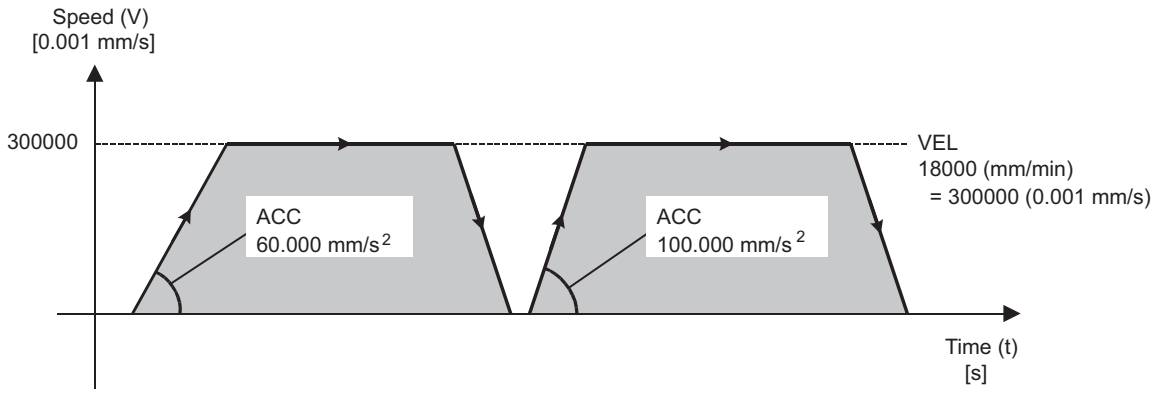


Fig. 8.7 Acceleration Time Change Command (ACC) Programming Example 2:
When Acceleration/Deceleration Degree Unit Selection is Set to 0 (Reference unit/s^2)

(5) Supplemental Information on ACC Command

(a) Related Motion Parameters

ACC changes the acceleration time of the following motion setting parameter.

Parameter Name	Register No.	Description
Straight Line Acceleration/ Acceleration Time Constant	OL□□36	Sets the linear acceleration rate or linear acceleration time constant.

The acceleration time can be changed by directly changing the setting of the motion setting parameter OL□□36 (Straight line acceleration time constant) instead of executing an ACC command. Refer to the following table for details on how to directly change the acceleration time setting.

Motion and setting procedure of acceleration time for positioning related commands are as follows.

Motion Module	Motion	Setting Procedure
SVA-01, PO-01, SVR	The axis moves according to the acceleration time of motion setting parameter OL□□36 (Straight line acceleration/ acceleration time constant).	Set an acceleration time to motion setting parameter OL□□36 (Straight line acceleration/acceleration time constant).
SVB-01, Built-in SVB	The axis moves at the acceleration rate of the SERVOPACK parameter.	Set an acceleration time using motion setting parameter OL□□36. Then, use bit 10 (ACC: Change acceleration time) of motion setting parameter OW□□08 (Motion command) to write the new acceleration time into the SERVOPACK. *

* The built-in SVB and SVB-01 modules have a function that can automatically write the setting of motion setting parameter OL□□36 (Straight line acceleration/acceleration time constant) into the SERVOPACK parameters. When this automatic writing function is enabled, you do not need to use bit 10 (ACC: Change acceleration time) of motion setting parameter OW□□08.
For the automatic writing function, refer to *11.6 Parameters That Are Automatically Updated* in *Machine Controller MP2000 Series Built-in SVB/SVB-01 Motion Module User's Manual* (manual number: SIEPC88070033).

(b) Setting Acceleration Time and Deceleration Time

With the following combination of motion module and SERVOPACK model, the acceleration time and deceleration time cannot be set individually. If you set the acceleration time, the deceleration time will be automatically set. With SERVOPACK models other than those listed below, the acceleration times and deceleration times can be set individually using ACC and DCC commands.

Motion Module	SERVOPACK Model	Description
SVB-01, Built-in SVB	SGD-N	• With a built-in SVB or SVB-01 module, the axis moves at the acceleration/deceleration rate of the SERVOPACK parameter. • SGD-N and SGDB-N SERVOPACKs uses same parameter to set both acceleration time and deceleration time.
	SGDB-N	

8.1.4 Deceleration Time Change (DCC)

Motion Programs	Sequence Programs
Applicable	Not applicable

(1) Overview

The Deceleration Time Change command (DCC) changes the deceleration time or deceleration rate of the axis for which one of the following axis move commands is executed.

- Positioning (MOV)
- Set Time Positioning (MVT)
- External Positioning (EXM)

The deceleration time for up to 16 axes can be changed at once. The deceleration time of any axis unspecified in the command block is not changed.
The deceleration time changed by the DCC command remains until it is reset by the next DCC command.

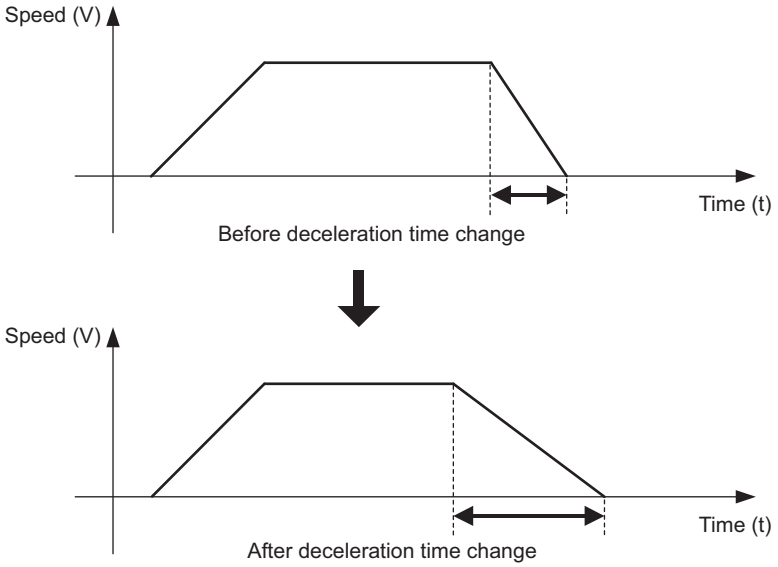


Fig. 8.8 Deceleration Time Change (DCC)

INFO

- The Deceleration Time Change command (DCC) changes the deceleration time for positioning related commands MOV, EXM, and MVT. Use the IDC command to set the interpolation related commands MVS, MCW, MCC, and SKP.
- All motion modules, except the PO-01 module, support ACC, DCC, and SCC commands.
  Use of these commands with the PO-01 module requires the following combination of MP2000-series Machine Controller and PO-01 module versions.

MP2000 Series Version Number	PO-01 Version Number
Ver.2.46 or later	Ver.1.06 or later

(2) Format

DCC [*Logical axis name 1*] *Deceleration time* [*Logical axis name 2*] *Deceleration time* [*Logical axis name 3*] *Deceleration time* . . . ;

Item	Unit	Data That Can Be Used
Deceleration time or Deceleration rate	ms or reference unit/s^2 Note: Use bits 4 to 7 of the motion setting parameter OW□□03 (Acceleration/deceleration degree unit selection) to set the unit.	· Directly specified value · Double integer register (Indirect designation)

(3) Setting Items for the DCC Command

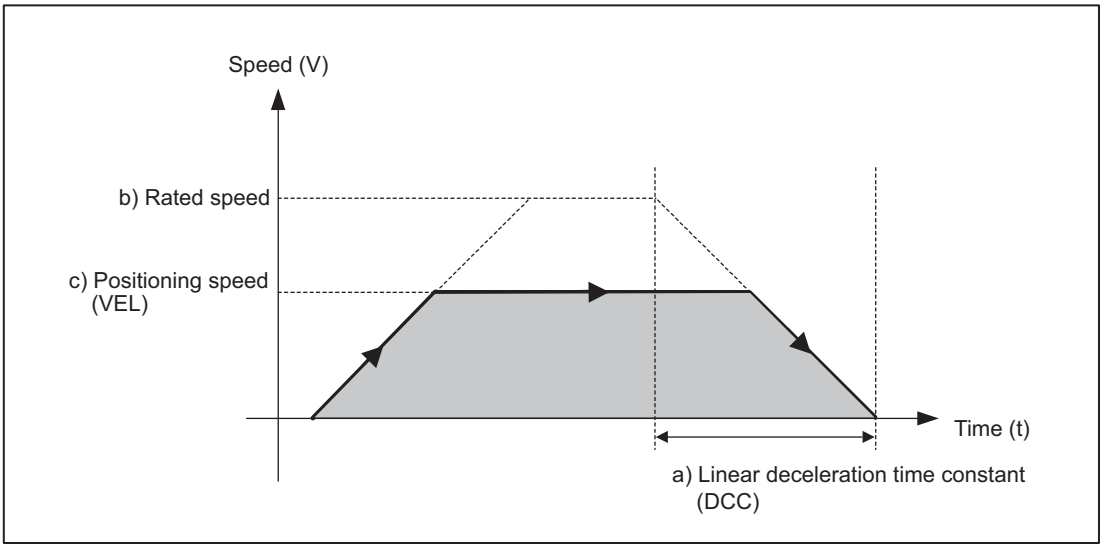
Either deceleration time (ms) or deceleration rate (reference unit/s^2) can be selected for the unit of set value for the DCC command.

Motion setting parameter OW□□03, bit 4 to 7 (Acceleration/deceleration degree unit selection)

Parameter Name	Acceleration/Deceleration Unit
Function setting 1, Acceleration/deceleration degree unit selection	0: Reference unit/s^2 1: ms (default)

- When bits 4 to 7 (Acceleration/deceleration degree unit selection) of OW□□03 are set to 1 (ms)

Motion Image



a) Linear deceleration time constant

Set the linear deceleration time constant (the time required to decelerate to 0 from the rated speed) for DCC command. The reference range is as follows.

1 to 32767 (ms)

b) Rated speed

Set the rated speed of each axis using motion fixed parameter 34 (Rated motor speed).
For details, refer to the manual for the motion module to be used.

c) Positioning speed

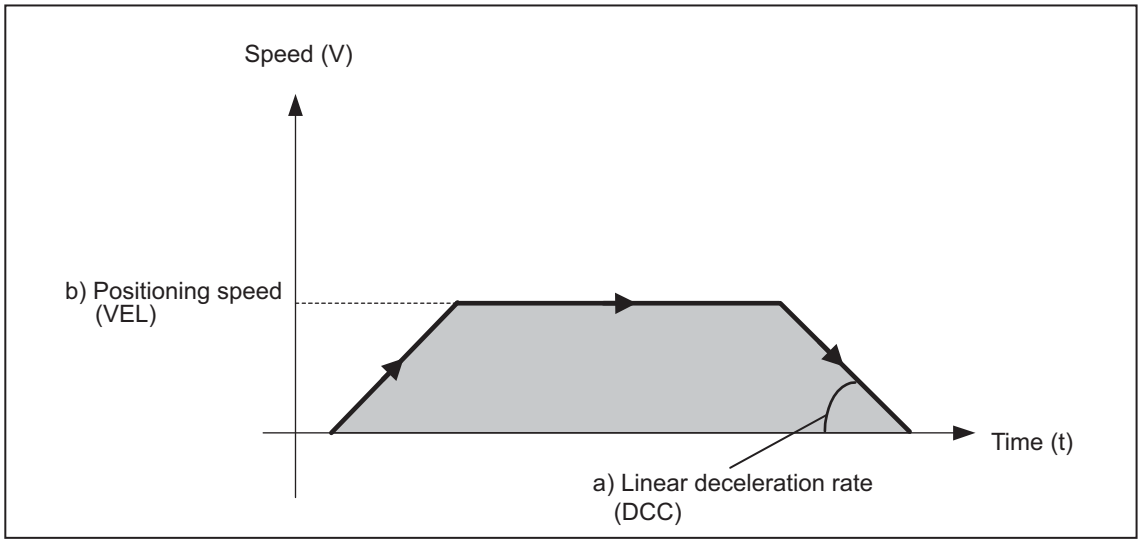
The speed for positioning related commands MOV, MVT, or EXM.
Set the positioning speed of each axis using the Set Velocity command (VEL).

INFO

For the Set Time Positioning command (MVT), the positioning speed is not the reference value of the VEL command. The Set Time Positioning command (MVT) changes the positioning speed according to the set positioning time and the amount of movement.

- When bits 4 to 7 (Acceleration/deceleration degree unit selection) of OW□□03 are set to 0 (reference unit/s^2)

Motion Image



a) Linear deceleration rate

Set the linear acceleration rate value for the DCC command.
The reference range is as follows.

1 to 2^{31}-1 (reference unit/s^2)

b) Positioning speed

The speed of positioning related value commands MOV, MVT, or EXM.
Set the positioning speed of each axis using the Set Velocity command (VEL).

INFO

For the Set Time Positioning command (MVT), the positioning speed is not the reference value of the VEL command. The Set Time Positioning command (MVT) changes the positioning speed according to the set positioning time and the amount of movement.

(4) Programming Examples

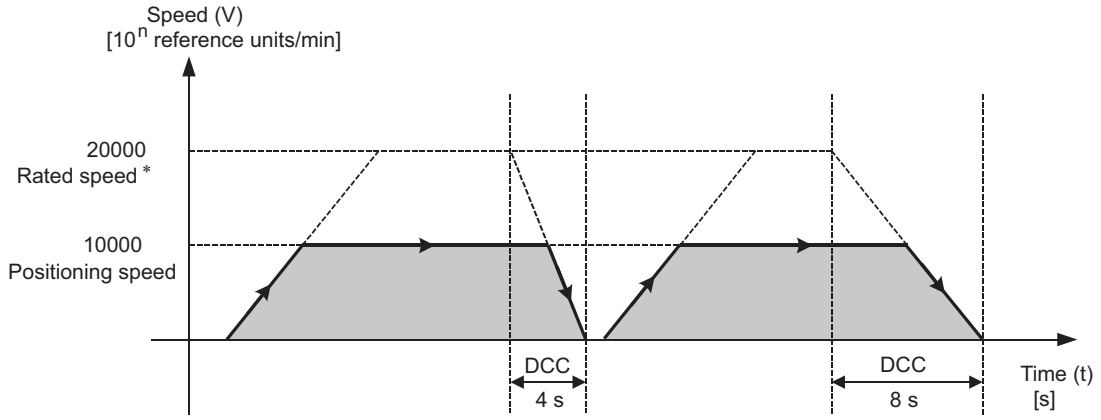
DCC command programming examples are shown below.

◀ EXAMPLE ▶

- When bits 4 to 7 (Acceleration/deceleration degree unit selection) of OW□□03 are set to 1 (ms).

The following example shows when the MOV command is executed to decelerate axis A1 from the rated speed to 0 in four seconds, and when the MOV command is executed to decelerate the axis from the rated speed to 0 in eight seconds.

```
INC;                  " Incremental mode
VEL [A1]10000;        " Set Velocity [10**n reference units/min]
ACC [A1]8000;         " Acceleration time change [ms]
DCC [A1]4000;         " Deceleration time change [ms]
MOV [A1]5000000;      " Positioning
DL00000 = 8000;       " Deceleration time [ms]
DCC[A1]DL00000;       " Deceleration time change [ms]
MOV [A1]5000000;      " Positioning
END;
```



* The unit of rated speed (min^{-1}) must be converted to the same unit used for positioning speed (10^n reference units/min).

Fig. 8.9 Deceleration Time Change (DCC) Command Programming Example 1: When Acceleration/Deceleration Degree Unit Selection is Set to 1 (ms).

◀ EXAMPLE ▶

- When bits 4 to 7 (Acceleration/deceleration degree unit selection) of OW□□03 are set to 0 (reference unit/s^2)

The following example shows when the MOV command is executed to decelerate axis A1 at the rate of 60.000 (mm/s^2) and when the MOV command is executed to decelerate the axis A1 at the rate of 100.000 (mm/s^2). In this example, one reference unit is 0.001 mm.

```
INC;                  " Incremental mode
VEL [A1]18000;        " Set Velocity [10**n reference units/min]
ACC [A1]100000;       " Acceleration rate change [reference unit//S*S]
DCC [A1]60000;        " Deceleration rate change [reference unit/S*S]
MOV [A1]5000000;      " Positioning
DL00000 = 100000;     " Deceleration rate [reference unit/S*S]
DCC [A1] DL00000;     " Deceleration rate change [reference unit/S*S]
MOV [A1]5000000;      " Positioning
END;
```

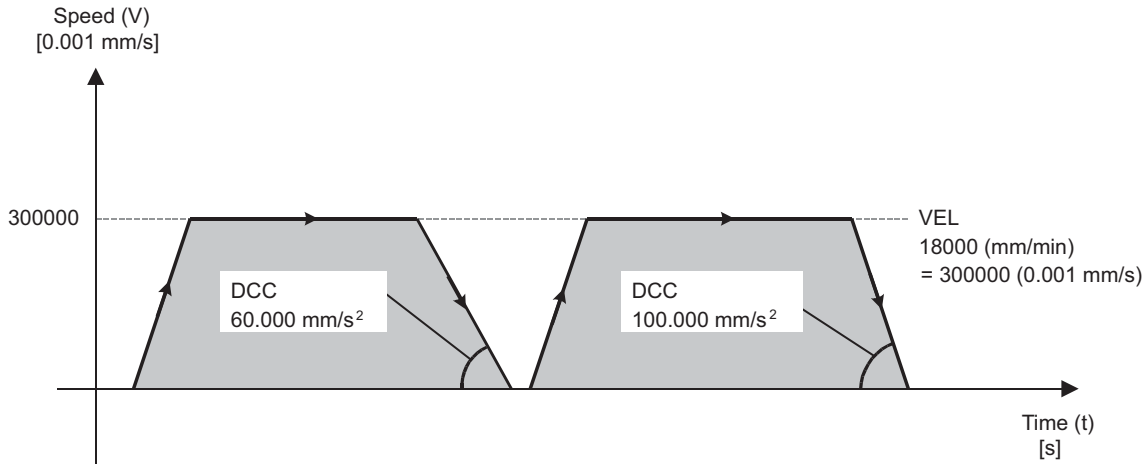


Fig. 8.10 Deceleration Time Change Command (DCC) Programming Example 2:
When Acceleration/Deceleration Degree Unit Selection is Set to 0 (Reference unit/s^2)

(5) Supplemental Information on DCC Command

(a) Related Motion Parameters

DCC changes the deceleration time of the following motion setting parameter.

Parameter Name	Register No.	Description
Straight Line Deceleration/ Deceleration Time Constant	OL□□38	Sets the linear deceleration rate or linear deceleration time constant.

The deceleration time can be changed by directly changing the setting of the motion setting parameter OL□□38 (Straight line deceleration/deceleration time constant) instead of executing an DCC command. Refer to the following table for details on how to directly change the deceleration time setting.

Motion and setting procedure of acceleration time for positioning related commands are as follows.

Motion Module	Motion	Setting Procedure
SVA-01, PO-01, SVR	The axis moves according to the deceleration time of motion setting parameter OL□□38 (Straight line deceleration/ deceleration time constant).	Set a deceleration time to motion setting parameter OL□□38 (Straight line deceleration/deceleration time constant).
SVB-01, Built-in SVB	The axis moves at the deceleration rate of the SERVOPACK parameter.	Set a deceleration time using motion setting parameter OL□□38. Then, use bit 11 (ACC: Change deceleration time) of motion setting parameter OW□□08 (Motion command) to write the new acceleration time into the SERVOPACK. *

* The built-in SVB and SVB-01 modules have a function that can automatically write the setting of motion setting parameter OL□□38 (Straight line deceleration/deceleration time constant) into the SERVOPACK parameters. When this automatic writing function is enabled, you do not need to use bit 11 (ACC: Change deceleration time) of motion setting parameter OW□□08.
For the automatic writing function, refer to *11.6 Parameters That Are Automatically Updated* in *Machine Controller MP2000 Series Built-in SVB/SVB-01 Motion Module User's Manual* (manual number: SIEPC88070033).

(b) Setting Acceleration Time and Deceleration Time

With the following combination of motion module and SERVOPACK model, the acceleration time and deceleration time cannot be set individually. If you set the acceleration time, the deceleration time will be automatically set. With SERVOPACK models other than those listed below, the acceleration times and deceleration times can be set individually using ACC and DCC commands.

Motion Module	SERVOPACK Model	Description
SVB-01, Built-in SVB	SGD-N	• With a built-in SVB or SVB-01 module, the axis moves at the acceleration/deceleration rate of the SERVOPACK parameter. • SGD-N and SGDB-N SERVOPACKs uses same parameter to set both acceleration time and deceleration time.
	SGDB-N	

8.1.5 S-curve Time Constant Change (SCC)

Motion Programs	Sequence Programs
Applicable	Not applicable

(1) Overview

The S-curve Time Constant Change command (SCC) changes the S-curve time constant of each axis for which an axis move command is executed.
The S-curve time constant is the parameter for the S-curve acceleration/deceleration function that suppresses mechanical vibration during acceleration/deceleration.
The S-curve time constant for up to 16 axes can be changed at once. The S-curve time constant of any axis unspecified in the command block is not changed.
The S-curve time constant changed by the SCC command remains until it is reset by the next SCC command.

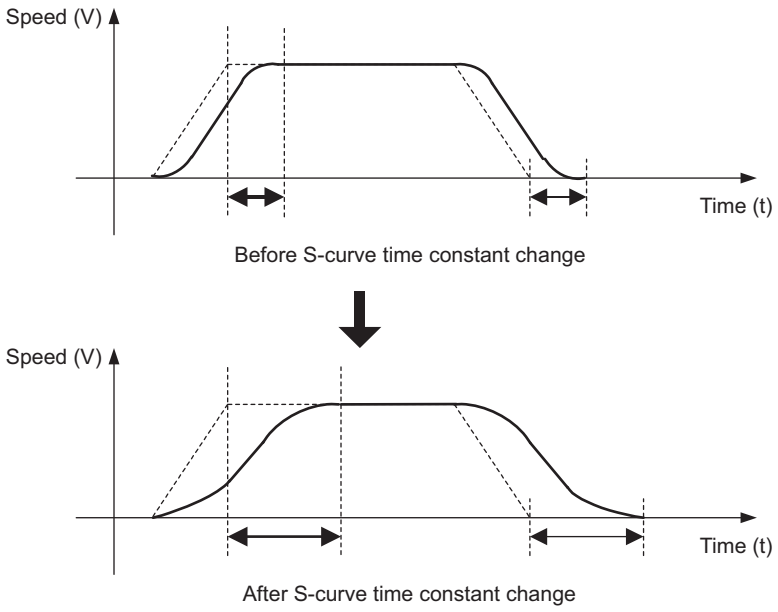


Fig. 8.11 S-curve Time Constant Change

INFO

All motion modules, except the PO-01 module, support ACC, DCC, and SCC commands.
Use of these commands with PO-01 module requires the following combination of Machine Controller and PO-01 module version numbers.

MP2000 Series Version Number	PO-01 Version Number
Ver.2.46 or later	Ver.1.06 or later

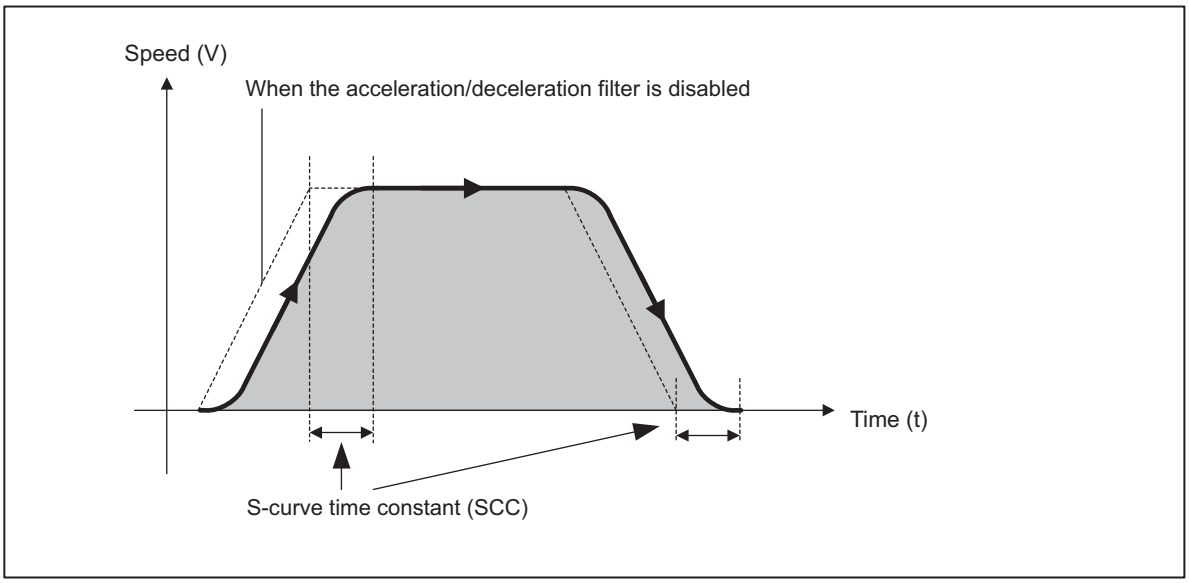
(2) Format

SCC [*Logical axis name 1*] *S-curve time constant* [*Logical axis name 2*] *S-curve time constant* . . . ;

Item	Unit	Data That Can Be Used
S-curve time constant	ms	· Directly designated value · Double integer register (indirect designation)

(3) Setting Items for SCC Command

Motion Image



Specify a numerical value or register for the S-curve time constant of each axis using an SCC command.
The reference range of the S-curve time constant depends on the motion module to be used:

- For SVA-01, PO-01, and SVR modules, the reference range is specified by motion setting parameter OW□□3A (Filter time constant).
- For built-in SVB and SVB-01 modules, the reference range is specified by the SERVOPACK parameter Average Movement Time.

The following table shows the reference range of S-curve time constants for each motion module model.

Motion Module	SCC Command Reference Range (ms)	Remarks
SVA-01	0 to 6553	–
SVB-01, Built-in SVB	0 to 510	With the SGD-N, SGDB-N, SGDH+NS110/NS115, SGDS, SGDX, or SGDV SERVOPACK
	–	With the SGDJ SERVOPACK, the S-curve acceleration/deceleration cannot be used since the SERVOPACK does not have a parameter for Average Movement Time.
PO-01	0 to 6553	–
SVR	0 to 6553	–

INFO

- If a reference value of more than 6553 ms is input, a motion program alarm will occur regardless of the motion module model.
- If a reference value exceeds the upper limit (511 to 6553 ms) when using a built-in SVB or SVB-01 module, bit 1 of motion monitoring parameter IL□□02 is set to 1 (Set parameter error), and the upper limit (510 ms) is set for the Average Movement Time of the SERVOPACK parameter.

(4) Programming Examples

◀ EXAMPLE ▶

An SCC command programming example is given below.
The following example shows when a MOV command with an S-curve time constant of 250 ms and a MOV command with an S-curve time constant of 500 ms are executed.
The parameters are set as follows for execution of this program.

- Motion setting parameter OW□□03, bits 0 to 3 (Speed unit selection) = 0 (reference unit/s)
- Motion setting parameter OW□□03, bits 4 to 7 (Acceleration/deceleration degree unit selection) = 0 (reference unit/s^2)

```
INC;                " Incremental mode
VEL [A1]10000;      " Set Velocity [reference unit/S]
ACC [A1]20000;      " Acceleration rate change [reference unit/S*S]
DCC [A1]20000;      " Deceleration rate change [reference unit/S*S]
SCC [A1]250;        " S-curve time constant change [ms]
MOV [A1]20000;      " Positioning
DL00000 = 500;      " S-curve time constant [ms]
SCC [A1]DL00000;    " S-curve time constant change [ms]
MOV [A1]20000;      " Positioning
END;
```

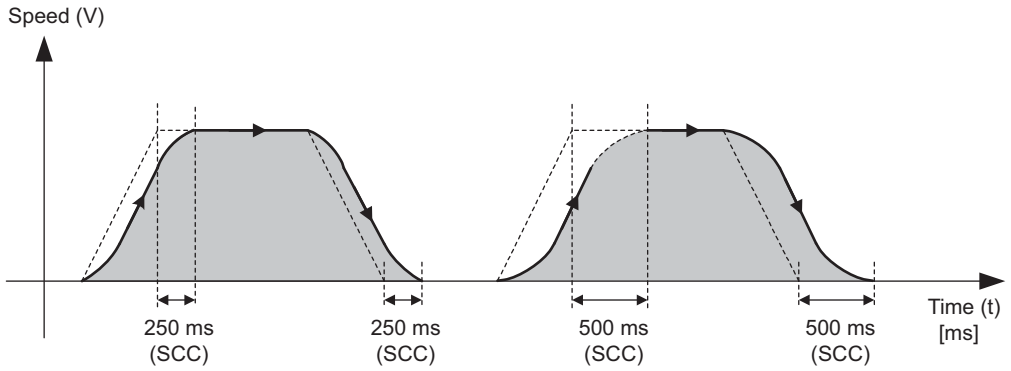


Fig. 8.12 S-curve Time Change Command Programming Example

(5) Supplemental Information on SCC Command

(a) Related Motion Parameters

SCC command changes the S-curve time constant of the following motion setting parameter.

Parameter Name	Register No.	Description
Filter time constant	OW□□3A	Sets the acceleration/deceleration filter time constant (1 = 0.1 ms). • Change the filter time constant after confirming that bit 0 of motion monitoring parameter IW□□0C is set to 1 (discharging completed). • Change the filter time constant after selecting a filter type using bits 8 to B of motion setting parameter OW□□03 (Filter type selection).

The S-curve time constant can be changed by directly changing the motion setting parameter OW□□3A (filter time constant) instead of executing an SCC command. Refer to the following table for information on how to directly change the S-curve time constant.

Motion and setting procedure of S-curve time constant are as follows.

Motion Module	Motion	Setting Procedure
SVA-01, PO-01, SVR	When S-curve acceleration/deceleration is enabled, the axis moves according to the S-curve time constant set in motion setting parameter OW□□3A (Filter time constant).	Set the S-curve time constant to motion setting parameter OW□□3A (filter time constant).
SVB-01, Built-in SVB	When the S-curve acceleration/deceleration is enabled, the axis moves according to the Average Movement Filter Time Constant of the SERVOPACK parameter.	Set the S-curve time constant using the motion setting parameter OW□□3A (filter time constant). Then, execute 12 (change filter time constant) of motion setting parameter OW□□08 (Motion command) to write the set S-curve time constant into the SERVOPACK.*

* The built-in SVB and SVB-01 modules can automatically write the setting of motion setting parameter OW□□3A into Average Movement Filter Time Constant of the SERVOPACK parameter.
When this automatic writing function is enabled, you do not need to execute 12 of motion setting parameter OW□□08.
For the automatic writing function, refer to *11.6 Parameters That Are Automatically Updated* in *Machine Controller MP2000 Series Built-in SVB/SVB-01 Motion Module User's Manual* (manual number: SIEPC88070033).

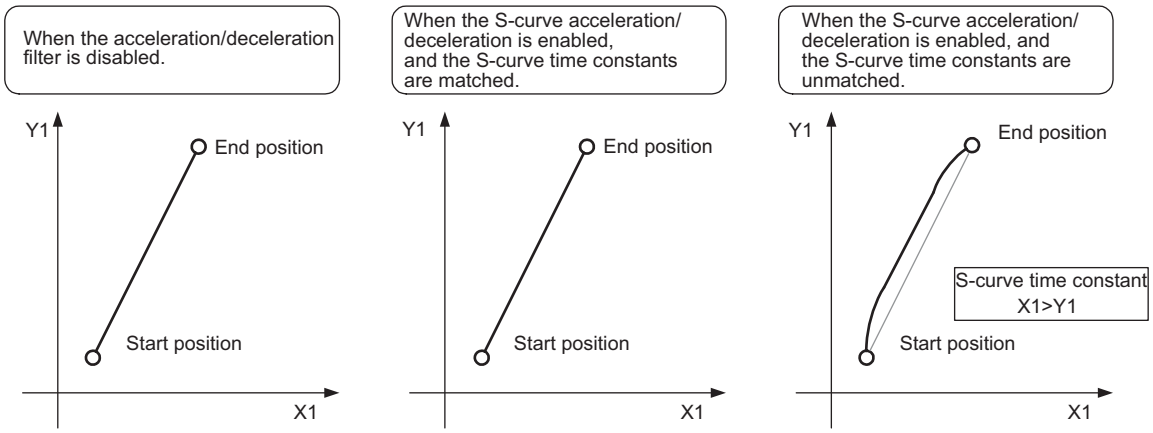
(b) Movement Path by Interpolation Command and S-Curve Acceleration/Deceleration

The S-curve acceleration/deceleration influences the movement path by using the interpolation related command MVS. MCW, MCC, or SKP.

- To achieve the same motion path as when the S-curve acceleration/deceleration is disabled for linear interpolation, set the same S-curve time constant for all the axes involved in interpolation.
- When the S-curve acceleration/deceleration is enabled for circular interpolation, the motion path will not be the same as when the S-curve acceleration/deceleration is disabled.

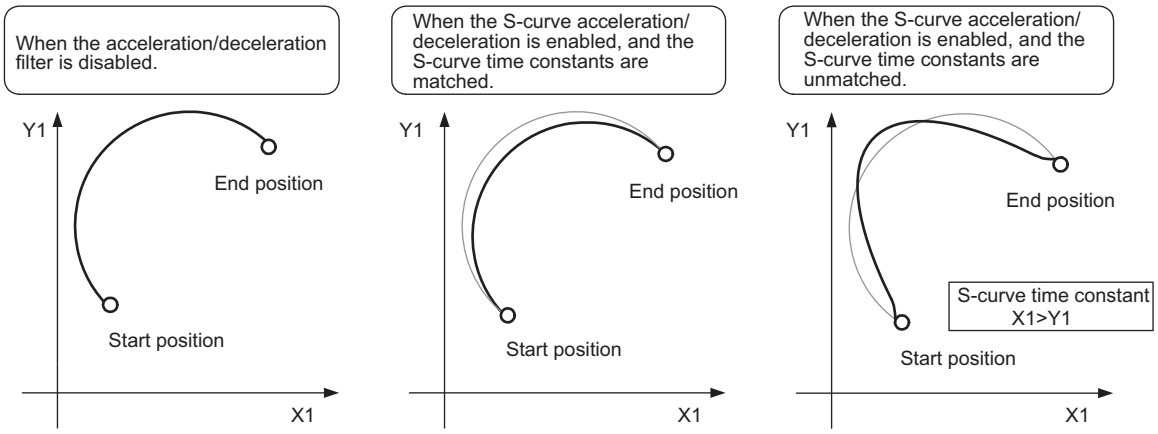
- Axis Movement Path by Linear Interpolation

◀ EXAMPLE ▶



- Axis Movement Path by Circular Interpolation

◀ EXAMPLE ▶



(c) Filter Type Selection

Before enabling the S-curve acceleration/deceleration, set the filter type of each axis by setting bit 8 to B (Filter type selection) of OW□□03 to 2: (Moving Average Filter).

Parameter Name	Register No.	Filter Type
Function setting 1, Filter type selection	OW□□03, bit 8 to B	0: No filter (default) 1: Exponential acceleration/deceleration filter 2: Moving average filter

When a built-in SVB or SVB-01 module is used and the function for automatically writing the parameters into the SERVOPACK is disabled, execute 13 (change filter type) of the motion setting parameter OW□□08 (Motion commands) to write the set filter type into the SERVOPACK.
An example of programming to change the filter type using a motion program is shown below.

Note: When using an SVA-01, PO-01, or SVR module, the following program is not necessary.
When using an SVB or SVB-01 module, the following program can be omitted by enabling the function to write parameters into the SERVOPACK.

```
:
:

" Verify if changing the filter type is allowed
IOW IW8008 == 0;              " Wait for response No motion command in execution
IOW IB800C0 == 1;             " Wait for response Discharging completed

" Select the filter type Moving Average Filter
DW00000 = OW8003 & F0FFH;   " Holds information other than Filter Type Selection
OW8003  = DW00000 | 0200H;  " Filter type = Moving average filter

" Write the filter type from the built-in SVB/SVB-01 module into the SERVOPACK
OW8008 = 13;                  " Request to change filter type
IOW IW8008 == 13;             " Wait for response Processing SCC
IOW IB80098 == 1;             " Wait for response Command execution completed
OW8008 = 0;                   " Clears the request
IOW IW8008 == 0;              " Wait for response No motion command in execution

:
:
```

INFO

Refer to *11.6 Parameters That Are Automatically Updated* in *Machine Controller MP2000 Series Built-in SVB/SVB-01 Motion Module User's Manual* (manual number: SIEPC88070033) for information on the built-in SVB/SVB-01 module function for automatically writing parameters into the SERVOPACK.

8.1.6 Set Velocity (VEL)

Motion Programs	Sequence Programs
Applicable	Not applicable

(1) Overview

The Set Velocity command (VEL) changes the feed speed of each axis for the following axis move commands.

- Positioning (MOV)
- External positioning (EXM)

In this manual, the above axis move commands and the Set Time Positioning command (MVT) are referred to as positioning related commands, and the feed speed of positioning related commands is referred to as positioning speed.
The positioning speed for up to 16 axes can be changed at once. The positioning speed of an axis unspecified in the VEL command block is not changed.
The axis positioning speed changed by a VEL command remains until it is reset by the next VEL command or it is changed by executing the Set Time Positioning command.

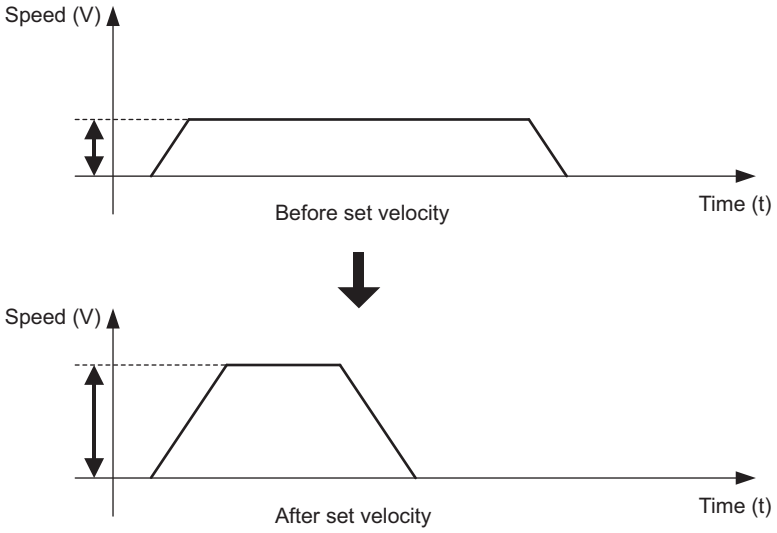


Fig. 8.13 Set Velocity (VEL)

INFO

The Set Velocity command (VEL) sets the positioning speed for positioning related commands (MOV and EXM). Use F designation or IFP command to set the feed speed for interpolation related commands (MVS, MCW, MCC, and SKP).

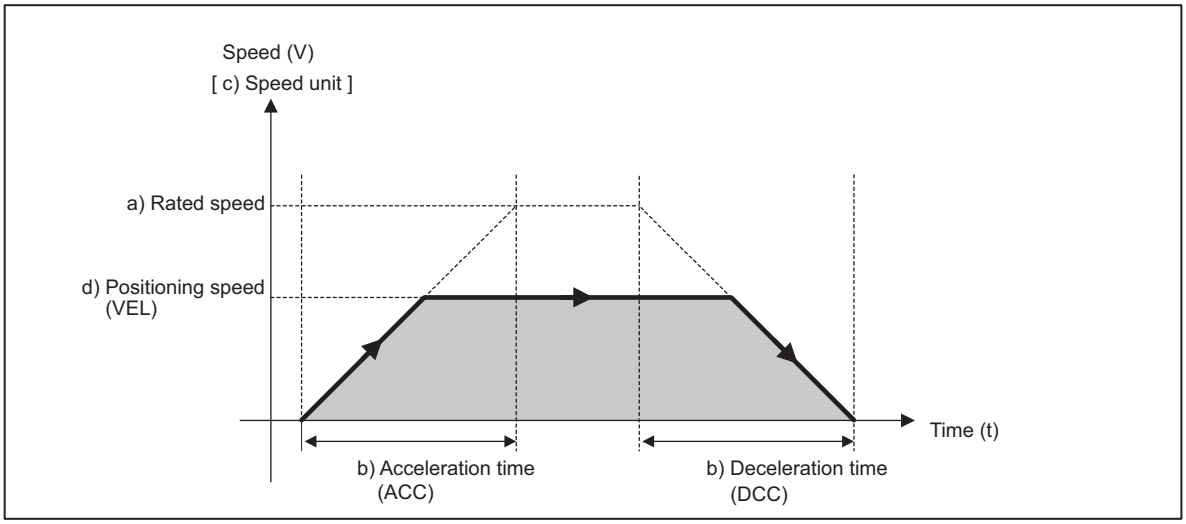
(2) Format

VEL [*Logical axis name 1*] *Positioning speed* [*Logical axis name 2*] *Positioning speed* ・・・ ;

Item	Unit	Usable Data
Positioning speed	10^n reference unit/min, Reference unit/s, 0.01% (percentage of rated speed), or 0.0001% (percentage of rated speed) Note: Use bits 0 to 3 (Speed unit selection) of motion setting parameter OW□□03 to select a setting unit.	・Directly specified value ・Double integer register (Indirect designation)

(3) Setting Items for VEL Command

Motion Image



a) Rated speed

Use motion fixed parameter 34 (Rated motor speed) to set the rated speed of each axis.
For details, refer to the manual for the motion module to be used.

b) Acceleration time/Deceleration time

Use the Acceleration Time Change (ACC)/Deceleration Time Change (DCC) command to set the acceleration/deceleration time for each axis.
The time set by executing ACC command is the time required to reach the rated speed.

c) Speed unit

Use bits 0 to 3 (Speed unit selection) of motion setting parameter OW□□03 to set the speed unit for each axis. The default is 10^n reference units/min.

Parameter Name	Register No.	Speed Unit	Reference Range
Function setting 1, Speed unit selection	OW□□03, bit 0 to 3	0: Reference units/s	0 to 2^{31}-1 (reference units/s)
		1:10^n reference units/min	0 to 2^{31}-1 (10^n reference units/min)
		2: 0.01% (percentage of rated speed)	0 to 32767 (0.01%)
		3: 0.0001% (percentage of rated speed)	0 to 3276700 (0.0001%)

INFO

The setting unit for VEL command when 10^n reference units/min is selected for the speed unit is determined by the motion fixed parameter 4: Reference unit selection.

Motion Fixed Parameter 4: Reference Unit Selection	Speed Unit: 10^n reference units/min	Remarks
pulse	1 = 1000 pulses/min	• When pulse is selected for reference unit: n = 3 • When a reference unit other than pulse is selected: n = Motion fixed parameter 5: Number of digits below decimal point
mm	1 = 1 mm/min	
deg	1 = 1 deg/min	
inch	1 = 1 inches/min	
μm	1 = 1μm/min	

d) Positioning speed

Specify a numerical value or register using a VEL command.

(4) Programming Examples

A VEL command programming example is shown below.
This example shows execution of a MOV command with the positioning speed specified as 40% of rated speed and a MOV command with the positioning speed specified as 20% of rated speed.

◀ EXAMPLE ▶

```
INC;                 " Incremental mode
ACC [A1]5000;        " Acceleration time change [ms]
DCC [A1]5000;        " Deceleration time change [ms]
VEL [A1]4000;        " Set Velocity [0.01%]
MOV [A1]3000000;     " Positioning
VEL [A1]2000;        " Set Velocity [0.01%]
MOV [A1]3000000;     " Positioning
END;
```

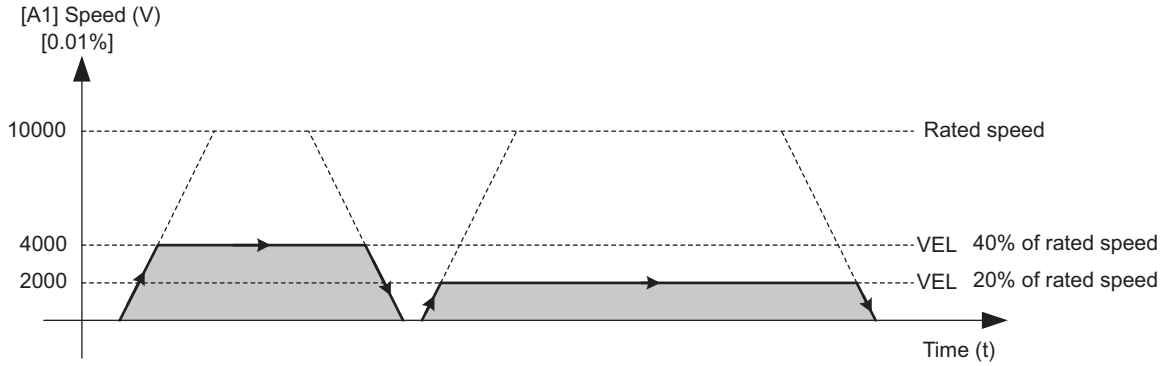


Fig. 8.14 Set Velocity Command (VEL) Programming Example
Speed unit: Percentage of rated speed in unit of 0.01%

(5) Supplemental Information on VEL Command

(a) Related Motion Parameters

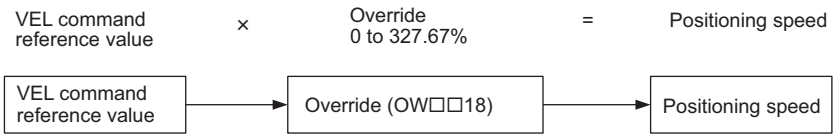
VEL command changes the positioning speed of the following motion setting parameter.

Parameter Name	Register No.	Description
Speed reference setting	OL□□10	Sets the speed reference value.

The positioning speed can be changed by directly changing the setting of motion setting parameter OL□□10 (Speed reference setting) instead of executing the VEL command.

(b) Override

Motion setting parameter OW□□18 (Override) can be used to specify a percentage of positioning speed specified by a VEL command to be output (output ratio).
The default setting of motion setting parameter OW□□18 is 10000 (100.00%).



TERMS

■ Override

Override often means "to invalidate." In this manual, however, it should be taken to mean "changing" the set value.

Motion setting parameter OW□□18 (Override) can be changed while an axis is moving.

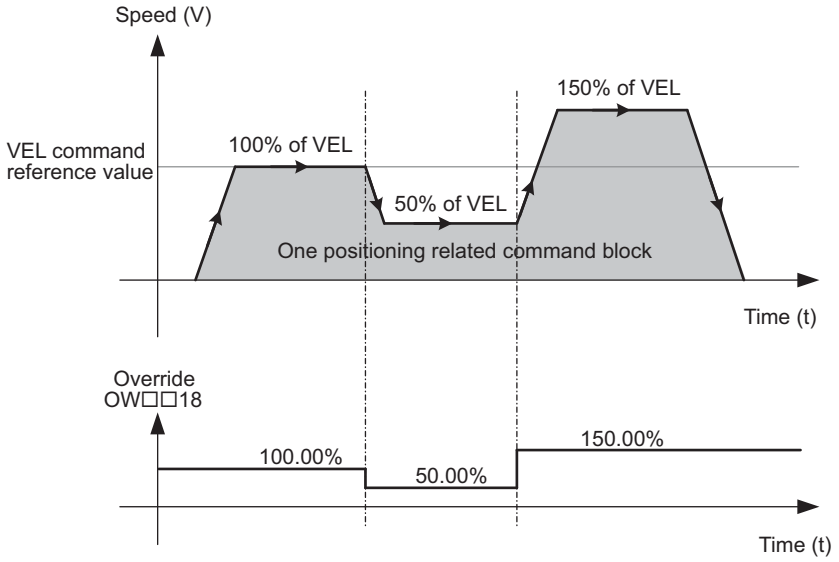


Fig. 8.15 OW□□18 (Override) and Positioning Related Command

INFO

- The SVR module does not have motion setting parameter OW□□18 (Override).
- For the Set Time Positioning command (MVT), the positioning speed used for override is not the VEL command reference value. The positioning speed changed by executing the Set Time Positioning command (MVT) is used for override.
- When using an override for the Set Time Positioning command (MVT), the positioning will not complete within the specified time.
  The MP2000 system calculates the positioning speed at execution of the Set Time Positioning command (MVT) assuming the override to be 100%.
- The speed unit of the rated speed specified by the motion fixed parameter is different from the speed unit used for VEL command coded in a motion program.

Speed	Speed Unit
Motion fixed parameter 34: Rated motor speed	rotations/min
Set Velocity command (VEL)	reference units/s, 10^n reference units/min, 0.01%, 0.0001%

Refer to *(c) Motor Speed Specifications* to calculate the rated speed according to the speed unit for the Set Velocity command (VEL).

(c) Motor Speed Specifications

In addition to the VEL command reference range, the motor rated speed and max. speed must be taken into consideration to decide the set value for the VEL command. To avoid causing an overspeed, check the speed specifications of your motor before setting a value for the VEL command.

INFO

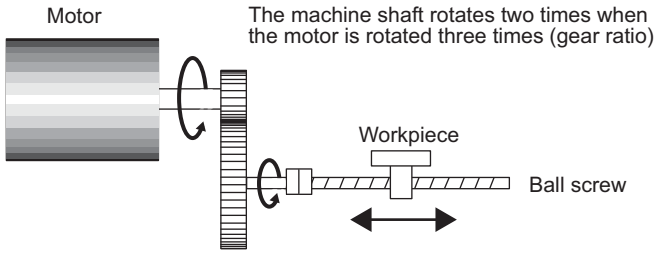
For rotational motors, the speed specifications are expressed in unit of rotations per specified time of period.

The rated speed when the speed unit is 10^n reference units/min is calculated according to the motion fixed parameter settings.

- Parameter Setting Example: When Electronic Gear is Enabled

  Note: When motion fixed parameter 4 (Reference unit selection) is set to a unit other than pulse, the electronic gear is valid.

Motion fixed parameters
- No. 4: Reference unit selection = mm
- No. 5: Number of digits below decimal point = 3
- No. 6: Travel distance per machine rotation = 10000 reference units
- No. 8: Servo motor gear ratio: 3
- No. 9: Machine gear ratio = 2
- No. 34: Rated motor speed = 3000 rotations/min



The table moves 10 mm for each rotation of the machine shaft
(Travel amount per one machine rotation)

When the electronic gear is enabled, n of the speed unit 10^n reference units/min is the number of digits below the decimal point. Therefore, the speed unit is:

10^n reference units/min = $10^3 \cdot$ 0.001mm/min = [mm/min]

The machine shaft rotation speed when the motor rotates at the rated speed:

Rated speed [rotations/min] $\times$ Gear ratio
= 3000 $\times$ (2/3) = 2000 [rotations/min]

To convert the number of rotations of the machine shaft into a reference unit (0.001 mm),

Travel distance per machine rotation [0.001 mm/rotation] $\times$ 2000 [rotation/min]
= 10000 $\times$ 2000 = 20000000 [0.001mm/min]

Because the speed unit is [mm/min],

20000000[0.001mm/min] = 20000[mm/min]

- Continues to the next page -

- Continued from the previous page -

- Parameter Setting Example: When the Electronic Gear is Disabled and an SVA-01 Module is Used

  Note: When motion fixed parameter 4 (Reference unit selection) is set to pulse, the electronic gear is invalid.

  Motion Fixed Parameters
  - No. 4: Reference unit selection = pulse
  - No. 22: Pulse counting mode selection = A/B × 4 (× 4)
  - No. 34: Rated motor speed = 3000 rotations per min
  - No. 36: Number of pulses per motor rotation (before multiplication) = 16384 pulses/rotation

When the electronic gear is disabled, n of the speed unit 10^n reference units/min is 3. Therefore, the speed unit is:

[10^n reference units/min] = [10^3 pulse/min] = [1000 pulses/min]

To convert the motor rated speed into pulses,

Rated motor speed [rotations/min] × (Number of pulses per motor rotation [pulse/rotation] × multiplier)
= 3000 × (16384 × 4) = 196608000 [pulses/min]

Because the speed unit is 1000 pulses/min,

196608000 [pulses/min] = 196608 [1000 pulses/min]

- Parameter Setting Example: When the Electronic Gear is Disabled and a Built-in SVB, SVB-01, PO-01, or SVR Module is Used

  Note: When motion fixed parameter 4 (Reference unit selection) is set to pulse, the electronic gear is invalid.

  Motion Fixed parameters
  - No. 4: Reference unit selection = pulse
  - No. 34: Rated motor speed = 3000 rotations/min
  - No. 36: Number of pulses per motor rotation = 65536 pulses/rotation

When the electronic gear is disabled, n of the speed unit 10^n reference units/min is 3. Therefore, the speed unit is:

[10^n reference units/min] = [10^3 pulses/min] = [1000 pulses/min]

To convert the motor rated speed into pulses,

Rated motor speed [rotations/min] × Number of pulses per motor rotation [pulses/rotation]
= 3000 × 65536 = 196608000 [pulses/min]

Because the speed unit is 1000 pulses/min,

196608000 [pulses/min] = 196608 [1000pulses/min]

Motion fixed parameters other than those mentioned in the examples may be required to be correctly set for the correct axis motions.
For details on each parameter and information on how to set them in accordance with the connected machine, refer to the relevant motion module manual.

8.1.7 Maximum Interpolation Feed Speed Setting (FMX)

Motion Programs	Sequence Programs
Applicable	Not applicable

(1) Overview

The Maximum Interpolation Feed Speed Setting command (FMX) sets the maximum feed speed for interpolation related commands MVS, MCW, MCC, and SKP. The maximum speed set by FMX command execution remains in effect until it is reset by the next FMX command.
The maximum feed speed for interpolation is not yet set at the moment the program operation starts. It is necessary to execute an FMX command before executing any of the following interpolation related commands.

- Linear Interpolation command (MVS)
- Circular Interpolation command (MCW, MCC)
- Helical Interpolation command (MCW, MCC)
- SKP command (Linear interpolation with skip function) (SKP)
- Interpolation Feed Speed Ratio Setting command (IFP)
- Interpolation Acceleration Time Change command (IAC)
- Interpolation Deceleration Time Change command (IDC)

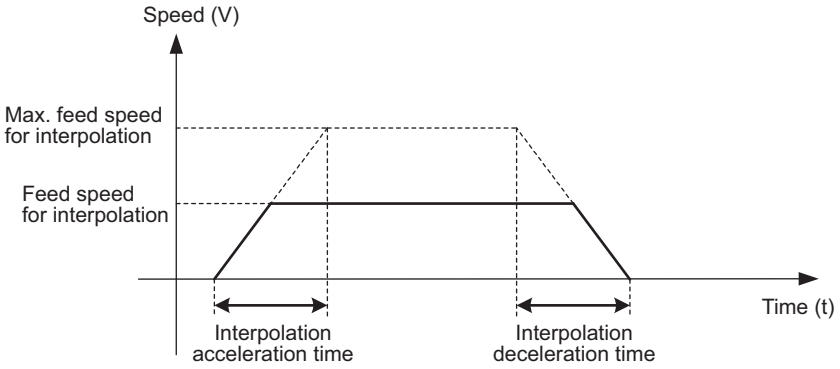


Fig. 8.16 Maximum Interpolation Feed Speed Setting Command (FMX)

INFO

- Execution of any interpolation related command is subject to the preset maximum feed speed for interpolation. For example, the Interpolation Acceleration Time Change command (IAC) and the Interpolation Deceleration Time Change (IDC) set the time required to reach the maximum feed speed for interpolation from the speed of 0.
- Executing an interpolation related command MVS, MCW, MCC, SKP, IFP, IAC, or IDC without first having executed an FMX command will cause an alarm in the motion program.

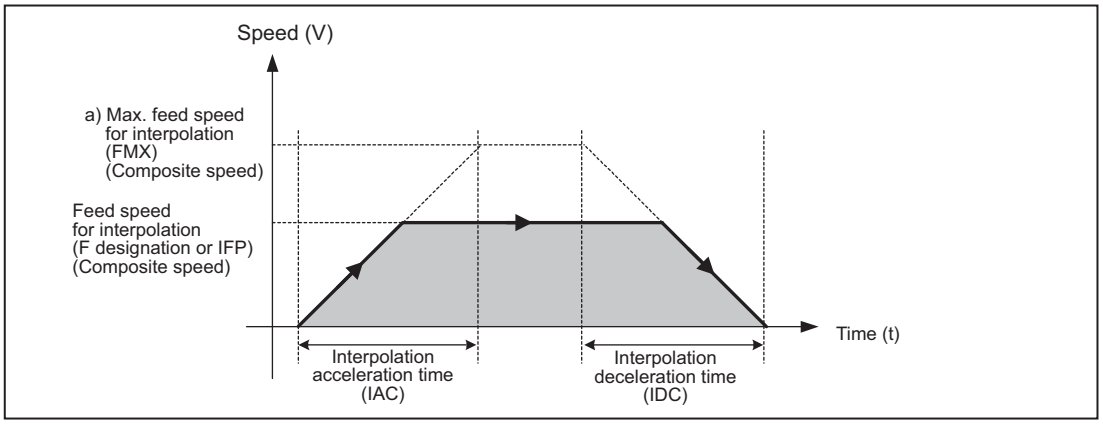
(2) Format

FMX *T Interpolation max. feed speed* ;

Item	Unit	Usable Data
Max. feed speed for interpolation	Reference units/min	· Directly designated value · Double integer type register (Indirect designation)

(3) Setting Items for FMX Command

Motion Image



a) Max. feed speed for interpolation

Specify a numerical value or register following to character T in the FMX command block. The maximum feed speed reference range is as follows.

1 to 2^{31}-1 (reference units/min)

The maximum feed speed for interpolation is the control data used for all interpolation related commands. Therefore, an FMX command must be coded at the beginning of the motion program when using an interpolation command MVS, MCW, MCC, or SKP.

(4) Programming Examples

An FMX command programming example is shown below.

◀ EXAMPLE ▶

```
INC;                              " Incremental mode
FMX T300000;                      " Maximum interpolation feed speed
IAC T4000;                        " Interpolation acceleration time change [ms]
IDC T4000;                        " Interpolation deceleration time change [ms]
IFP P75;                          " Interpolation feed speed ratio setting [%]
MVS [A1]30000 [B1]30000;          " Linear interpolation
MVS [A1]30000 [B1]30000 F150000;  " Linear interpolation (F command)
END;
```

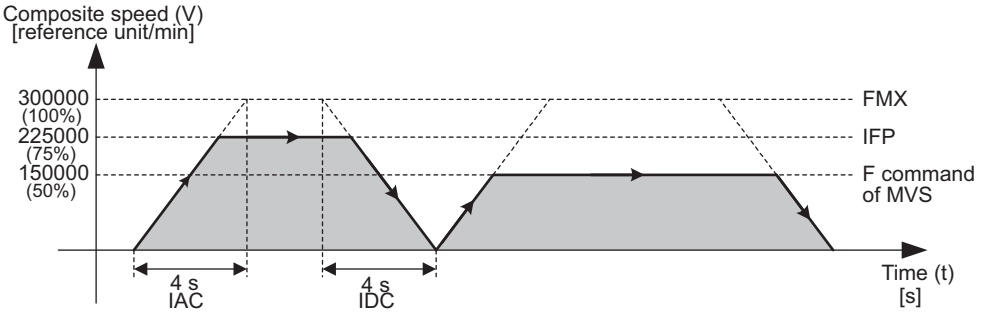


Fig. 8.17 Maximum Interpolation Feed Speed Command Programming Example

(5) Supplemental Information on FMX Command

(a) Related Motion Parameters

There are no motion setting parameters related to the FMX command.
The maximum feed speed for interpolation specified by executing an FMX command is the control data exclusively reserved for motion programs, and cannot be specified by using a motion setting parameter.

8.1.8 Interpolation Feed Speed Ratio Setting (IFP)

Motion Programs	Sequence Programs
Applicable	Not applicable

(1) Overview

The Interpolation Feed Speed Ratio Setting command (IFP) sets the feed speed for the following axis move commands as a percentage of the maximum feed speed for interpolation.

- Linear Interpolation command (MVS)
- Circular Interpolation command (MCW, MCC)
- Helical Interpolation command (MCW, MCC)
- SKP command (Linear interpolation with skip function) (SKP)

In this manual, the above axis move commands are referred to as Interpolation Related Commands, and the feed speed of Interpolation Related Commands is referred to as the Interpolation Feed Speed. The interpolation feed speed set by IFP command execution remains in effect until it is reset by the next IFP command.
The interpolation feed speed is not set when the program operation starts. It is necessary to set an interpolation feed speed by executing the Interpolation Feed Speed Ratio Setting (IFP) or F designation before executing any interpolation related commands.

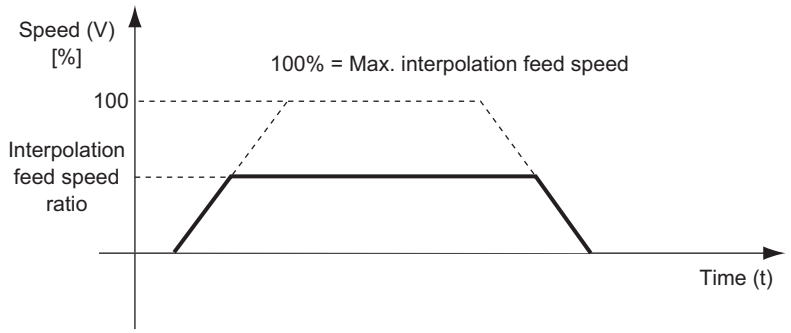


Fig. 8.18 Interpolation Feed Speed Ratio Setting Command (IFP)

INFO

- Specify the maximum feed speed for interpolation by executing the Maximum Interpolation Feed Speed Setting command (FMX) before executing the IFP command. Executing the IFP command without first having specified the maximum feed speed will cause an alarm in the motion program.
- F designations can be used to specify the interpolation feed speed by writing a numerical value or register following to a character F for interpolation related commands. Specify the interpolation feed speed in reference units/min.
- If an IFP command is executed after F designation, the interpolation feed speed specified by the F designation will be cancelled. If an F designation is executed after IFP command execution, the interpolation feed speed specified by the IFP command will be cancelled as well.
- Executing an interpolation related command without first having specified the interpolation feed speed will cause an alarm in the motion program.
- The Interpolation Feed Speed Ratio Setting command (IFP) sets the feed speed for interpolation related commands MVS, MCW, MCC, and SKP. Use the VEL command to set the feed speed for positioning related commands MOV and EXM.

(2) Format

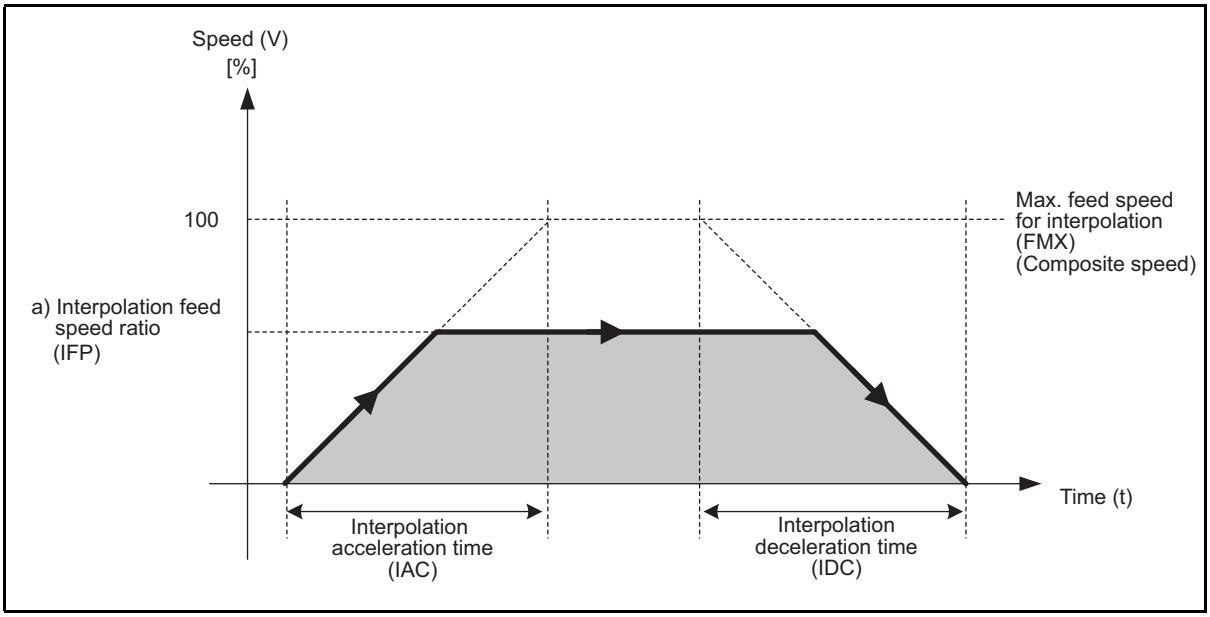
IFP *P Interpolation feed speed ratio*;

Item	Unit	Usable Data
Interpolation feed speed ratio	%	· Directly designated value · Double integer type register (Indirect designation)

INFO IFP command cannot be coded in the same block used for an interpolation command MVS, MCW, MCC, or SKP.

(3) Setting Items for IFP Command

Motion Image



a) Interpolation feed speed ratio

Specify a numerical value or register following to character P in the IFP command block.
The IFP command sets a percentage of the maximum interpolation feed speed (FMX) as the interpolation feed speed.
The interpolation feed speed is a composite speed of all the axes specified for an interpolation related command MVS, MCW, MCC, or SKP.
The interpolation feed speed ratio reference range is as follows.

1 to 100 (%)

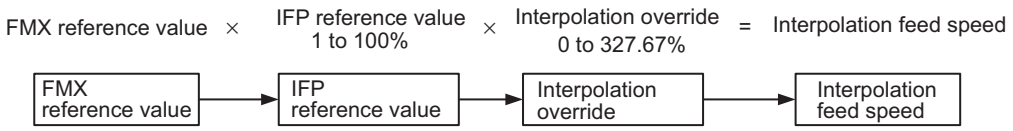
You can select whether or not to apply an interpolation override to the interpolation feed speed.
Refer to *4.3.3 Work Registers* for information on how to use the interpolation override.

- When not using an interpolation override

FMX reference value × IFP reference value 1 to 100% = Interpolation feed speed

FMX reference value → IFP reference value → Interpolation feed speed

• When using an interpolation override



INFO

- The interpolation feed speed can be specified by using either an IFP command or F designation. Refer to *b) Interpolation feed speed* in *(3) Setting Items for MVS Command* in *8.2.2 Linear Interpolation (MVS)* for details.
- If a value higher than 100 (%) is specified for the IFP reference value (%), an alarm will occur in the motion program.
- If an interpolation override applied interpolation feed speed exceeds the FMX reference value, the actual output value of the interpolation feed speed will be reset to the FMX reference value.

(4) Programming Examples

An IFP command programming example is shown below.

◀ EXAMPLE ▶

```
INC;                        " Incremental mode
FMX T300000;                " Maximum interpolation feed speed setting [reference unit/min]
IAC T4000;                  " Interpolation acceleration time change [ms]
IDC T4000;                  " Interpolation deceleration time change [ms]
IFP P75;                    " Interpolation feed speed ratio setting [%]
MVS [A1]30000 [B1]30000;    " Linear interpolation
DL00000 = 50;               " Interpolation feed speed ratio [%]
IFP PDL00000;               " Interpolation feed speed ratio setting [%]
MVS [A1]30000 [B1]30000;    " Linear interpolation
END;
```

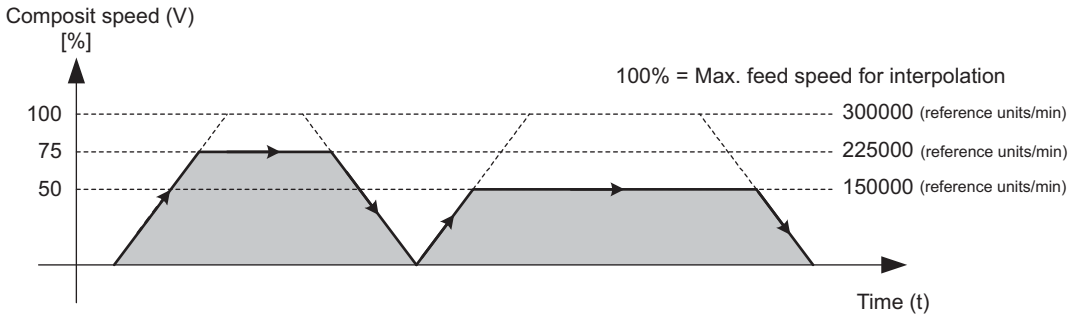


Fig. 8.19 Interpolation Feed Speed Ratio Setting Command (IFP)

(5) Supplemental Information on IFP Command

(a) Related Motion Parameters

There is no motion setting parameter related to the IFP command.
The interpolation feed speed ratio specified by executing an IFP command is the control data exclusively reserved for motion programs, and cannot be specified by using a motion setting parameter.

8.1.9 Interpolation Acceleration Time Change (IAC)

Motion Programs	Sequence Programs
Applicable	Not applicable

(1) Overview

The Interpolation Acceleration Time Change command (IAC) changes the acceleration time for the following axis move commands.

- Linear Interpolation command (MVS)
- Circular Interpolation command (MCW, MCC)
- Helical Interpolation command (MCW, MCC)
- SKP command (Linear interpolation with skip function) (SKP)

In this manual, the above axis move commands are referred to as interpolation related commands.
The Maximum Interpolation Feed Speed Setting command (FMX) must be executed before executing the IAC command. The acceleration time changed by the IAC command remains until it is reset by the next IAC command.
The interpolation acceleration time is 0 ms when the program starts running.

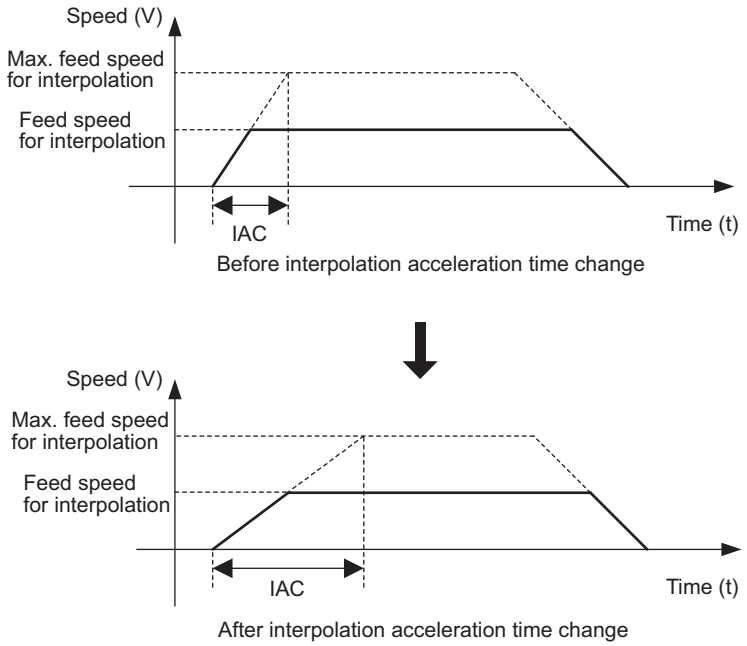


Fig. 8.20 Interpolation Acceleration Time Change Command (IAC)

INFO

The Interpolation Acceleration Time Change command (IAC) changes the acceleration time for interpolation related commands MVS, MCW, MCC, and SKP.
Use ACC command to set the acceleration time for positioning related commands MOV, EXM, and MVT.

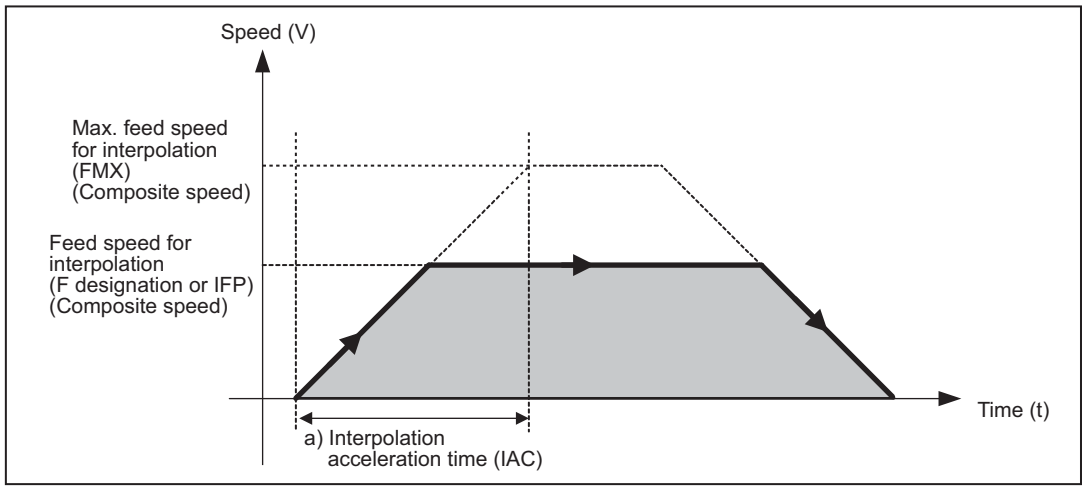
(2) Format

IAC *T Interpolation acceleration time*;

Item	Unit	Usable Data
Interpolation acceleration time	ms	· Directly designated value · Double integer type register (Indirect designation)

(3) Setting Items for IAC Command

Motion Image



a) Interpolation acceleration time

Specify a numerical value or register following to character T in the IAC command block.
The time set by executing an IAC command is the time required to reach the maximum feed speed from the speed of 0.
The interpolation acceleration time reference range is as follows.

0 to 32767 (ms)

(4) Programming Examples

An IAC command programming example is shown below.

◀ EXAMPLE ▶

```
INC;                                  " Incremental mode
FMX T300000;                          " Maximum interpolation feed speed setting [reference unit/min]
IDC T4000;                            " Interpolation deceleration time change [ms]
IAC T2000;                            " Interpolation acceleration time change [ms]
MVS [A1]30000 [B1]30000 F150000;      " Linear interpolation
DL00000 = 4000;                       " Interpolation acceleration time [ms]
IAC TDL00000;                         " Interpolation acceleration time change [ms]
MVS [A1]30000 [B1]30000;              " Linear interpolation
END;
```

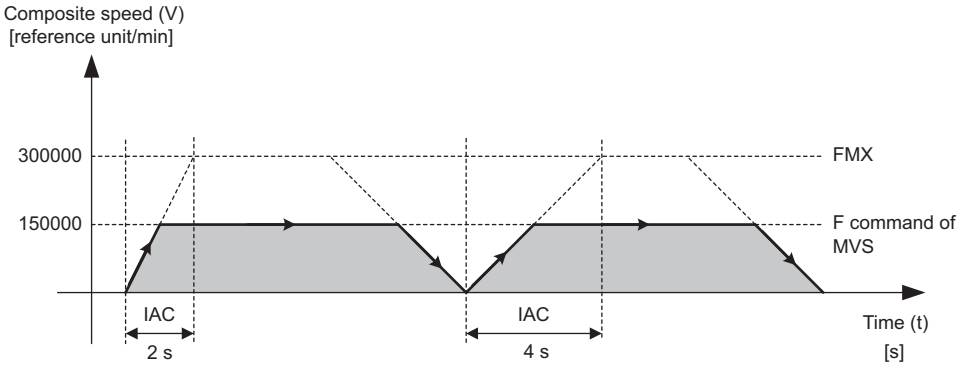


Fig. 8.21 Interpolation Acceleration Time Change Command Programming Example

(5) Supplemental Information on IAC Command

(a) Related Motion Parameters

There is no motion setting parameter related to the IAC command.
The interpolation acceleration time specified by executing the IAC command is the control data exclusively reserved for motion programs, and cannot be specified by using a motion setting parameter.

8.1.10 Interpolation Deceleration Time Change (IDC)

Motion Programs	Sequence Programs
Applicable	Not applicable

(1) Overview

The Interpolation Deceleration Time Change command (IDC) changes the deceleration time for the following axis move commands.

- Linear Interpolation command (MVS)
- Circular Interpolation command (MCW, MCC)
- Helical Interpolation command (MCW, MCC)
- SKP command (Linear interpolation with skip function) (SKP)

In this manual, the above axis move commands are referred to as interpolation related commands.
The Maximum Interpolation Feed Speed Setting command (FMX) must be executed before executing the IDC command. The deceleration time changed by the IDC command remains until it is reset by the next IDC command.
The interpolation deceleration time is 0 ms when the program starts running.

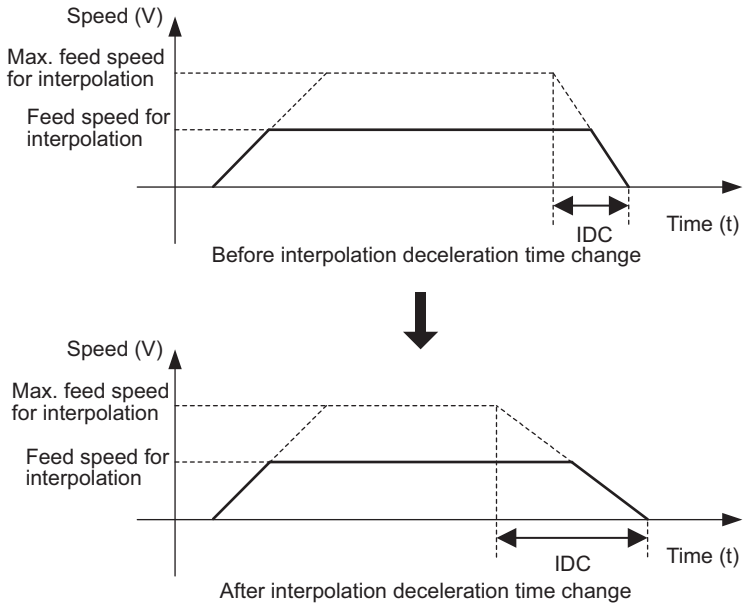


Fig. 8.22 Interpolation Deceleration Time Change Command (IDC)

INFO

The Interpolation Deceleration Time Change command (IDC) sets the deceleration time for interpolation related commands MVS, MCW, MCC, and SKP.
Use the DCC command to set the deceleration time for positioning related commands MOV, EXM, and MVT.

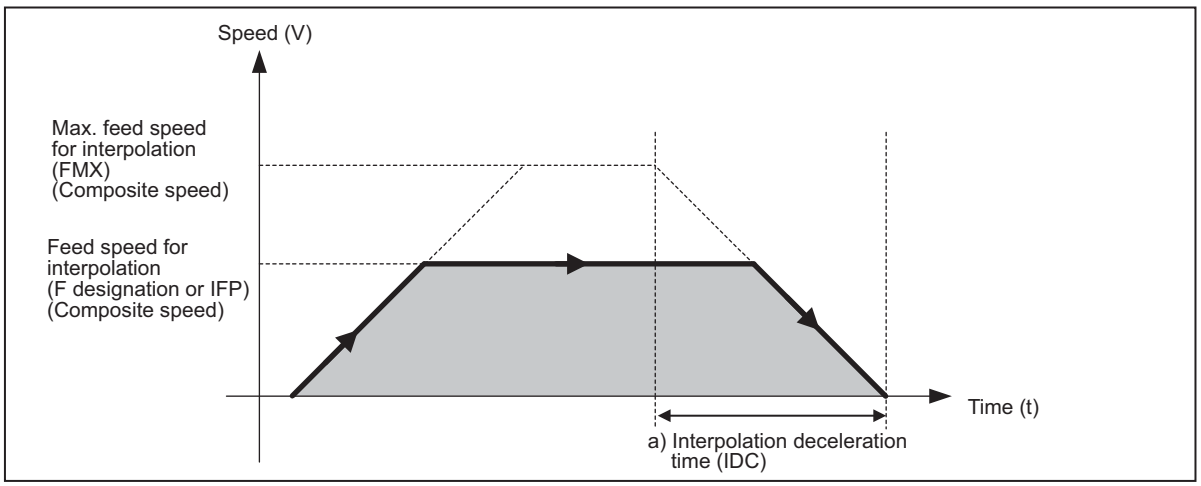
(2) Format

IDC *T Interpolation deceleration time* ;

Item	Unit	Usable Data
Interpolation deceleration time	ms	· Directly designated value · Double integer type register (Indirect designation)

(3) Setting Items for IDC Command

Motion Image



a) Interpolation deceleration time

Specify a numerical value or register following to T in the IDC command block.
The time set by executing the IDC command is the time to decelerate from the maximum feed speed for interpolation to the speed of 0.
The interpolation deceleration time reference range is as follows.

0 to 32767 (ms)

(4) Programming Examples

◀ EXAMPLE ▶

An IDC command programming example is shown below.

```
INC;                                  " Incremental mode
FMX T300000;                          " Maximum interpolation feed speed setting [reference uint/min]
IAC T4000;                            " Interpolation acceleration time change [ms]
IDC T2000;                            " Interpolation deceleration time change [ms]
MVS [A1]30000 [B1]30000 F150000;      " Linear interpolation
DL00000 = 4000;                       " Interpolation deceleration time [ms]
IDC TDL00000;                         " Interpolation deceleration time change [ms]
MVS [A1]30000 [B1]30000;              " Linear interpolation
END;
```

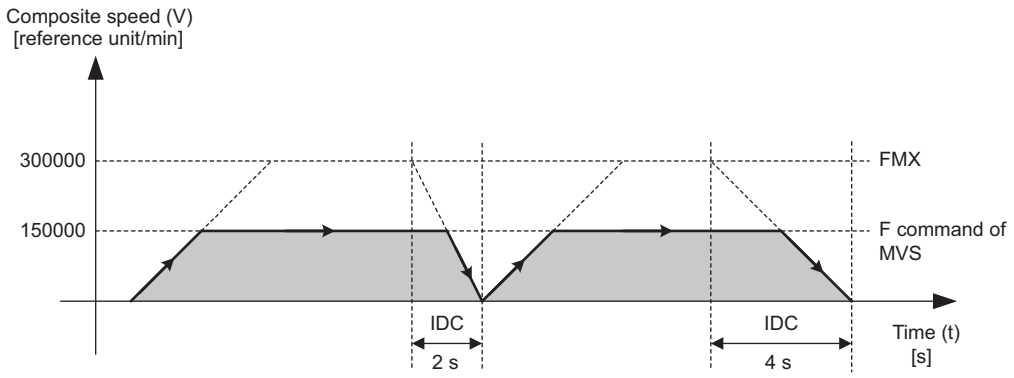


Fig. 8.23 Interpolation Deceleration Time Change Command Programming Example

(5) Supplemental Information on IDC Command

(a) Related Motion Parameters

There is no motion setting parameter related to the IDC command.
The interpolation deceleration time set by executing the IDC command is the control data exclusively reserved for motion programs, and cannot be set by using a motion setting parameter.

8.2 Axis Move Commands

This chapter described axis move commands.

8.2.1 Positioning (MOV)

Motion Programs	Sequence Programs
Applicable	Not Applicable

(1) Overview

The Positioning (MOV) command independently moves each axis from the current position to the end position at positioning speed.

Up to16 axes can be moved simultaneously. Any axis not specified in the command will not be moved.

The path of movement with the MOV command is different from the linear travel.

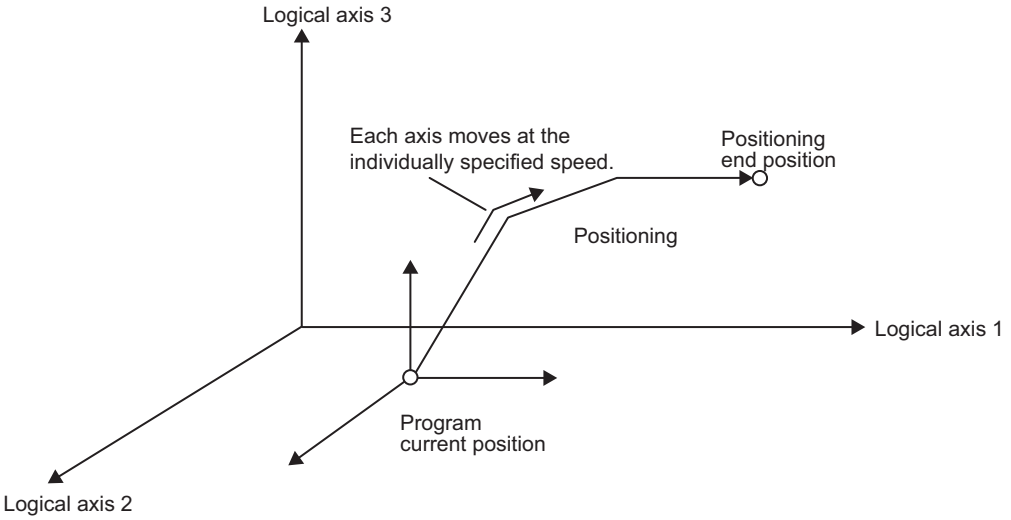


Fig. 8.24 Movement Path with MOV Command

⚠ CAUTION

- The path of movement with the Positioning (MOV) command is not always a straight line. When programming, be sure to check the path to make sure that there are no tools or other obstacles in the way of the workpiece.
  Failure to carry out this check may result in damage to equipment, serious personal injury, or even death.

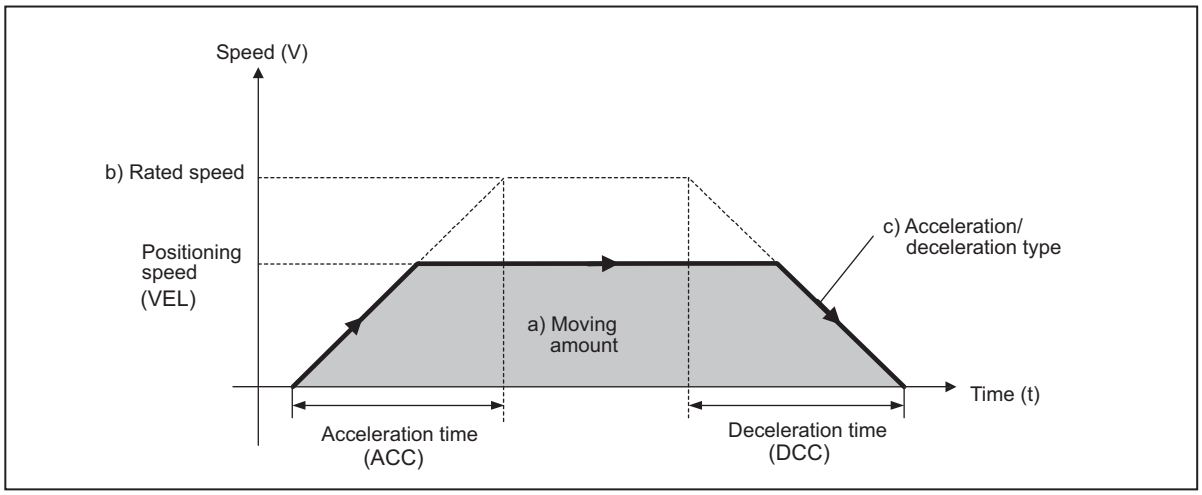
(2) Format

MOV [*Logial axis name 1*] *Reference position* [*Logical axis name 2*] *Reference position* [*Logical axis name 3*] *Reference position* ... ;

Item	Unit	Usable Data
Reference position	Reference unit	· Directly designated value · Double integer type register (Indirect designation)

(3) Setting Items for MOV Command

Motion Image



a) Moving amount

The moving amount of each axis differs depending on the movement mode, ABS mode or INC mode.

- Moving amount in ABS mode

  The difference between the program current position and the reference position.

- Moving amount in INC mode

  The reference position is used as the moving amount.

INFO Refer to *7.2.2 Reference Unit* for information on the moving amount unit.

b) Rated speed

Use motion fixed parameter 34 (Rated motor speed) to set the rated speed of each axis.
For details, refer to the relevant motion module manual.

c) Acceleration/deceleration type

The acceleration/deceleration for a MOV command can be selected from the following three types.
It can be selected according to the combination of the following settings:

Motion setting parameter OW□□03, bit 4 to 7: Acceleration/deceleration degree unit selection

Motion setting parameter OW□□03, bit 8 to B: Filter type selection

(a) No acceleration/deceleration

Movement with both the acceleration time and deceleration time set to 0

Setting Method	Movement Path
• OW□□03, bit 4 to 7: Acceleration/deceleration degree unit selection = 1 (ms) • OW□□03, bit 8 to B: Filter type selection = 0 (No filter) • Set the ACC command to 0. • Set the DCC command to 0.	

(b) Single-step linear acceleration/deceleration

Movement with a constant acceleration/deceleration speed

Setting Method	Movement Path
• OW□□03, bit 4 to 7: Acceleration/deceleration degree unit selection = 1 (ms) • OW□□03, bit 8 to B: Filter type selection = 0 (No filter) • Set the ACC command to a value other than 0. • Set the DCC command to a value other than 0.	

(c) S-curve acceleration/deceleration

Movement at S-curved acceleration/deceleration speed

Setting Method	Movement Path
• OW□□03, bit 4 to 7: Acceleration/deceleration degree unit selection = 1 (ms) • OW□□03, bit 8 to B: Filter type selection = 2 (Moving average filter) • Set the ACC command to a value other than 0. • Set the DCC command to a value other than 0. • Set the SCC command to a value other than 0.	

INFO

For axis movement with a MOV command, an in-position check is executed to verify that the axis enters the positioning completion range. After the in-position check, the next move command block will be executed.
The following diagram illustrates the in-position execution.

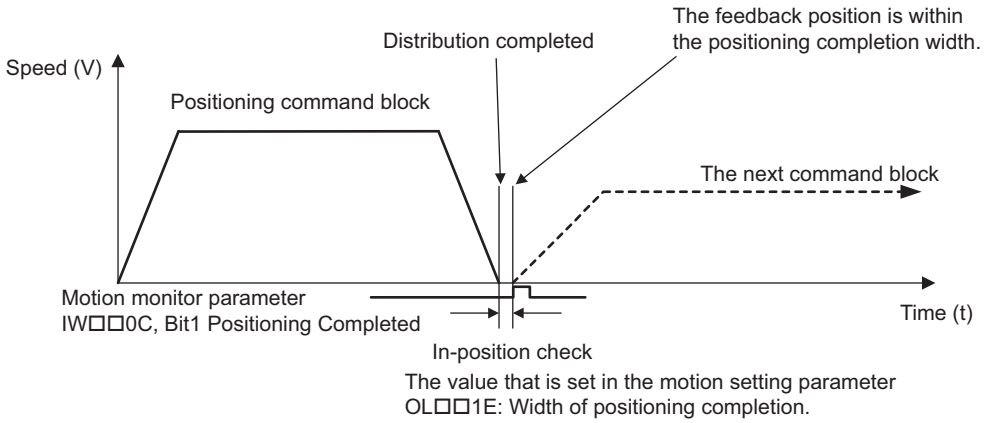


Fig. 8.25 Execution of In-position Check

(4) Programming Examples

A programming example for a MOV command in ABS mode is shown below.

◀ EXAMPLE ▶

```
ABS;
ACC [A1]1000 [B1]1000 [C1]1000;
DCC [A1]1000 [B1]1000 [C1]1000;
VEL [A1]2000 [B1]2000 [C1]2000;
MOV [A1]4000 [B1]3000 [C1]2000;
END;
```

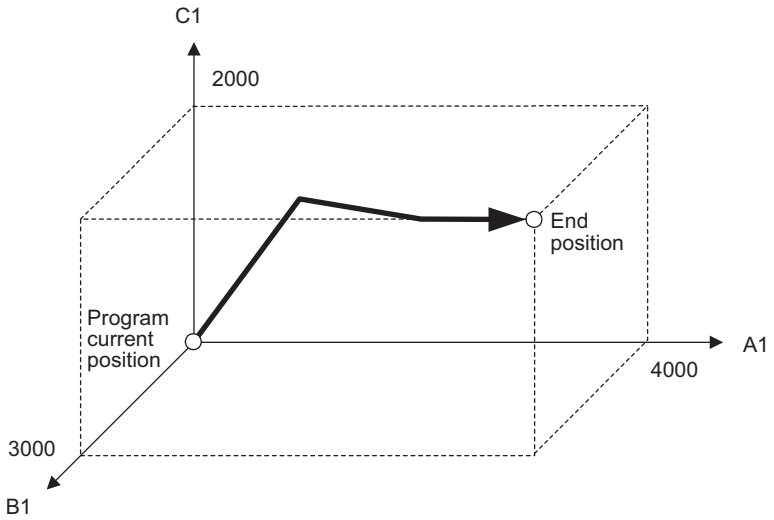


Fig. 8.26 MOV Command Programming Example

8.2.2 Linear Interpolation (MVS)

Motion Programs	Sequence Programs
Applicable	Not applicable

(1) Overview

The Linear Interpolation command (MVS) moves each axis on a straight line from the program current position to the end position at the specified interpolation feed speed. Up to 16 axes can be moved simultaneously. Any axis not specified in the command will not be moved.

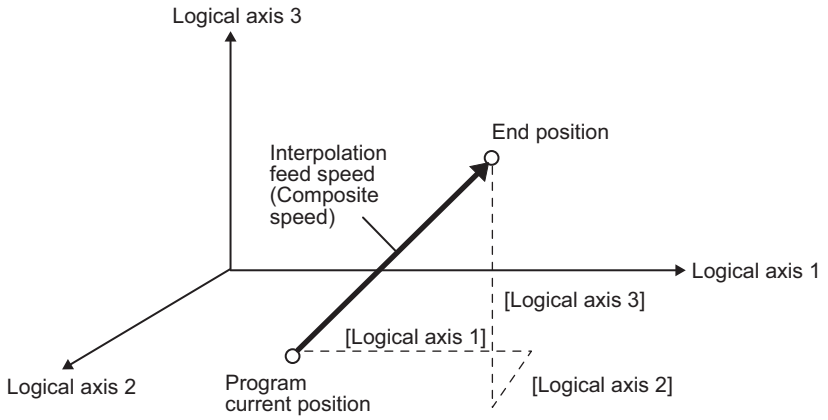


Fig. 8.27 Movement Path with MVS Command

⚠ CAUTION

- Linear Interpolation (MVS) can be executed for either linear axes or rotary axes. If rotary axes are included, however, the linear interpolation path will not be in straight line. When programming, be sure to check the path to make sure that there are no tools or other obstacles in the way of the workpiece.
  Failure to carry out this check may result in damage to equipment, serious personal injury, or even death.

INFO For axis movement with the MVS command, an in-position check is not automatically executed. Use the PFN command to execute an in-position check if required.

(2) Format

MVS [*Logical axis name 1*] *Reference position* [*Logical axis name 2*] *Reference position* [*Logical axis name 3*] *Reference position* ...

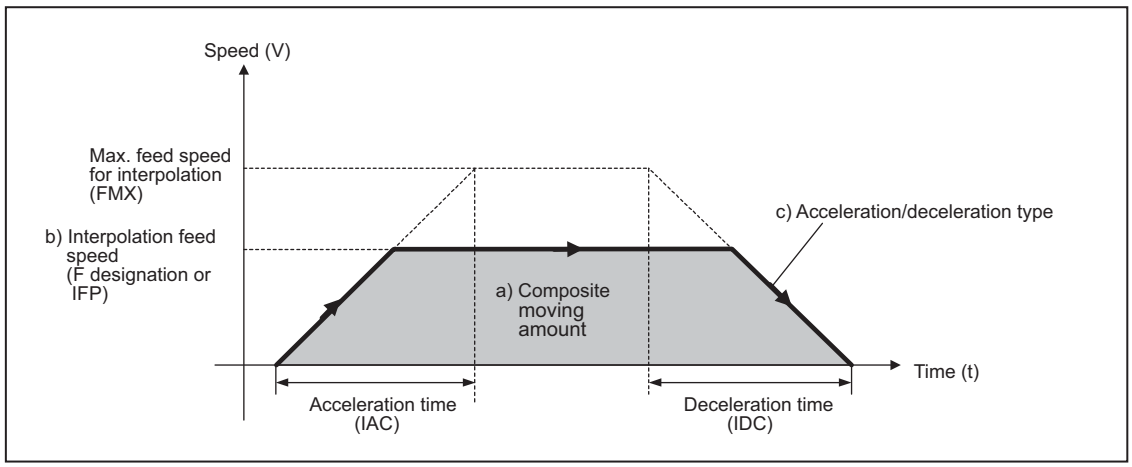
F Interpolation feed speed ;

Item	Unit	Usable Data
Reference position	Reference unit	· Directly designated value
Interpolation feed speed	Reference units/min	· Double integer type register (Indirect designation)

Note: The interpolation feed speed can be omitted.

(3) Setting Items for MVS Command

Motion Image



a) Composite moving amount

The composite moving amount differs depending on the movement mode: ABS mode or INC mode.

- Composite moving amount in ABS mode

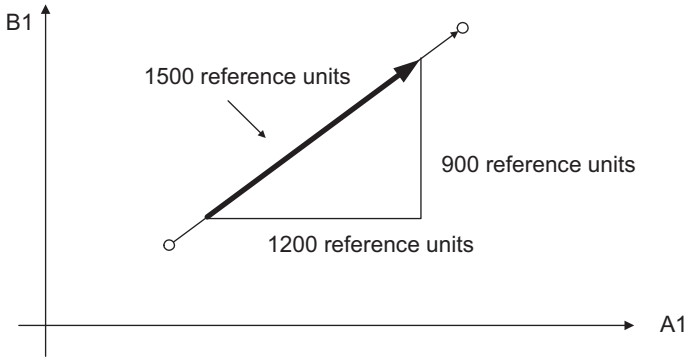
  The difference between the program current position and the reference position

- Composite movement amount in ABS mode

  The reference position is used as the moving amount.

INC command block: *INC MVS[A1]1200 [B1]900;*

Composite moving amount $= \sqrt{1200^2 + 900^2} = 1500$ (reference unit)



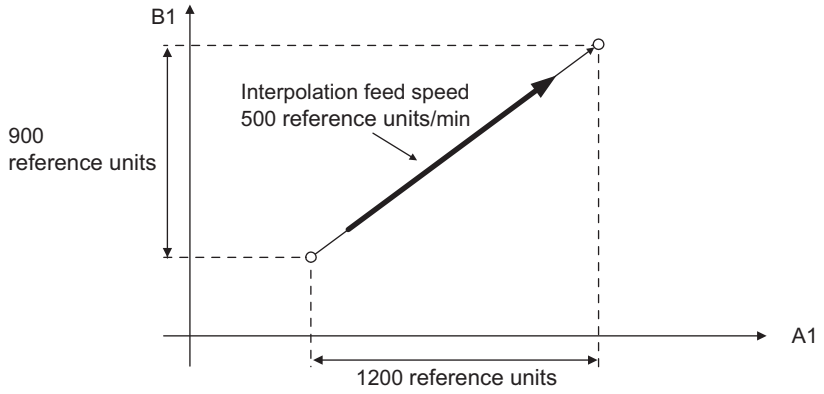
INFO

Refer to *7.2.2 Reference Unit* for information on the moving amount unit.

b) Interpolation feed speed (F command or IFP)

Specify a numerical value or register following to character F in the MVS command block (F designation). The specified interpolation feed speed is treated as a composite speed for all the specified axes.
The reference range is between 1 reference unit/min and the maximum feed speed for interpolation (FMX) [reference units/min].

For the INC command block INC MVS[A1]1200 [B1]900 F500;



INFO

The feed speed of each axis is calculated using the following formula.

The feed speed of each axis [reference units/min]

$$= \frac{\text{moving amount of each axis [reference units]}}{\text{composite moving amount [reference units]}} \times \text{interpolation feed speed [reference units/min]}$$

For example, the feed speed of each axis in above condition is calculated as following.

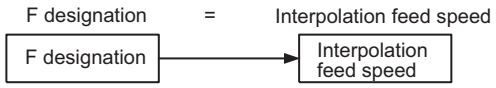
Interpolation feed speed (the value of F) = 500 [reference units/min]

Composite moving amount = $\sqrt{1200^2+900^2}$ = 1500 [reference units]

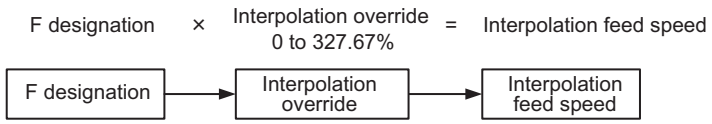
- The feed speed of A1 axis = $\frac{1200}{1500} \times 500 = 400$ [reference units/min]
- The feed speed of B1 axis = $\frac{900}{1500} \times 500 = 300$ [reference units/min]

You can select whether or not to apply an interpolation override to the F designation.
Refer to *4.3.3 Work Registers* for information on how to use an interpolation override.

- When not using an interpolation override



- When using an interpolation override



The interpolation feed speed can also be specified as a percentage of the maximum feed speed for interpolation (FMX).
Refer to *8.1.8 Interpolation Feed Speed Ratio Setting (IFP)* for information how to specify an interpolation feed speed as a percentage of the maximum feed speed for interpolation.

INFO

- If a value higher than the FMX reference value (reference units/min) is specified for an F designation (reference units/min), a motion alarm will occur.
- If the interpolation override applied interpolation feed speed exceeds the FMX value, the output value of the interpolation feed speed will be reset to the FMX reference value.
- When the interpolation feed speed is not specified in the command block, the interpolation feed speed specified in the previous command block will be applied.

The interpolation override can be changed during axis movement.

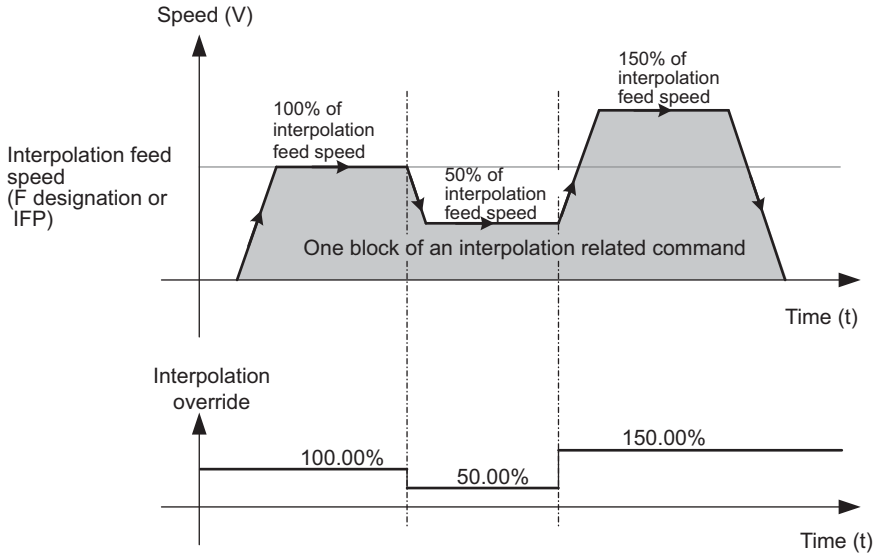


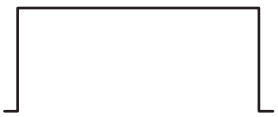
Fig. 8.28 Interpolation Override and Interpolation Related Command

c) Acceleration/deceleration type

The acceleration/deceleration for an MVS command can be selected from the following three types. It can be selected according to the combination of settings of IAC, IDC, and SCC commands and motion parameter OW□□03, bit 8 to B: Filter type selection.

(a) No acceleration/deceleration

Movement with both the acceleration time and deceleration time set to 0

Setting Method	Movement Path
• OW□□03, bit 8 to B: Filter type selection = 0 (No filter) • Set the IAC command to 0. • Set the IDC command to 0.	

(b) Single-step linear acceleration/deceleration

Movement with a constant acceleration/deceleration speed

Setting Method	Movement Path
• OW□□03, bit 8 to B: Filter type selection = 0 (No filter) • Set the IAC command to a value other than 0. • Set the IDC command to a value other than 0.	

(c) S-curve acceleration/deceleration

Movement at S-curved acceleration/deceleration speed

Setting Method	Movement Path
• OW□□03, bit 8 to B: Filter type selection = 2 (Moving average filter) • Set the IAC command to a value other than 0. • Set the IDC command to a value other than 0. • Set the SCC command to a value other than 0.	

INFO

- Code a FMX command to specify the maximum feed speed for interpolation at the beginning of motion program. Otherwise, a motion program alarm will occur at execution of the MVS command.
- If the acceleration/deceleration time is not specified, the default time of 0 ms is applied.
- For axis movement with the MVS command, an in-position check is not automatically executed. Use the PFN command to execute an in-position check if required.

(4) Programming Examples

A programming example of an MVS command in ABS mode is shown below.

◀ EXAMPLE ▶

```
FMX T30000000;
ABS;
IAC T1000;
IDC T1000;
MVS [A1]4000 [B1]3000 [C1]2000 F50000;
END;
```

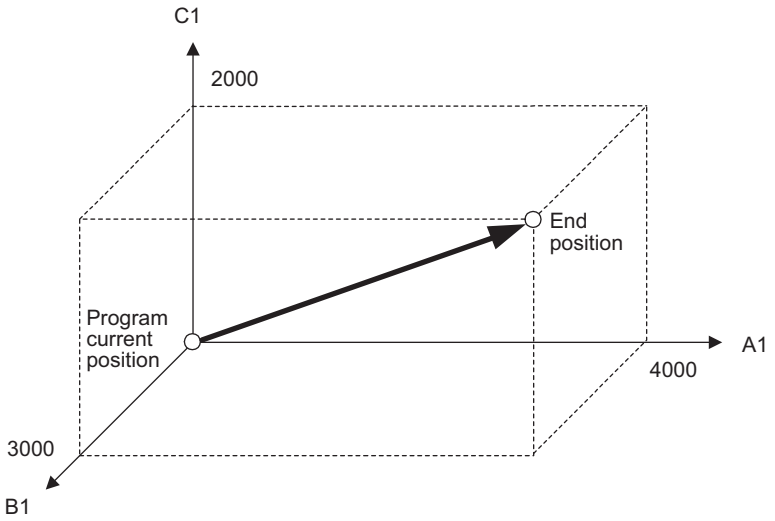


Fig. 8.29 MVS Command Programming Example

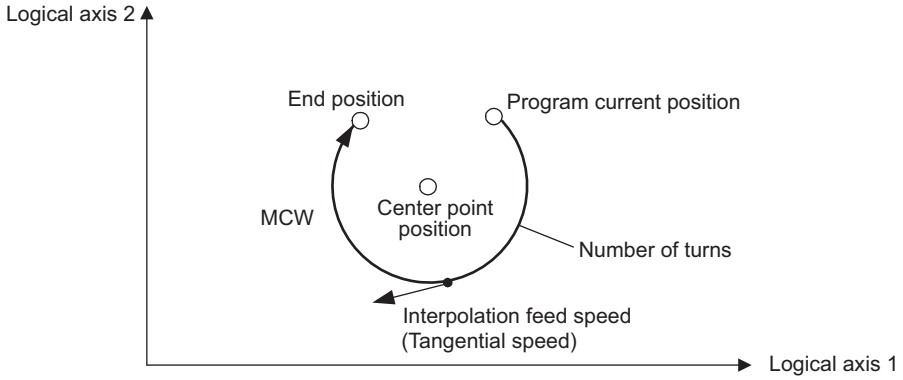
8.2.3 Clockwise/Counterclockwise Circular Interpolation with Center Position Designation (MCW, MCC)

Motion Programs	Sequence Programs
Applicable	Not applicable

(1) Overview

The Clockwise/Counterclockwise Circular Interpolation with Center Position Designation command (MCW, MCC) moves two axes simultaneously from the program current position to the end position on the designated plane at the interpolation feed speed on the circle determined by the center position.

- MCW: Circular Interpolation command for Clockwise (CW)
- MCC: Circular Interpolation command for Counterclockwise (CCW)



IMPORTANT

- Before designating a Clockwise/Counterclockwise Circular Interpolation command, be sure to designate the circular interpolation plane in Coordinate Plane Setting (PLN).
  If the PLN command is not coded, a motion program alarm will occur at execution of the MCW or MCC command.
- Designate the axes for the end position and center position in the same order as the axes are specified in the PLN command.
- Be sure to code the FMX command at the beginning of the program to specify the maximum feed speed for interpolation.
  Otherwise, a motion program alarm will occur at execution of the MCW or MCC command.
- When the acceleration/deceleration time is not specified, the default acceleration/deceleration time of 0 ms will be applied.

INFO

For axis movement with the MCW or MCC command, an in-position check is not automatically executed. Use the PFN command to execute an in-position check if required.

(2) Format

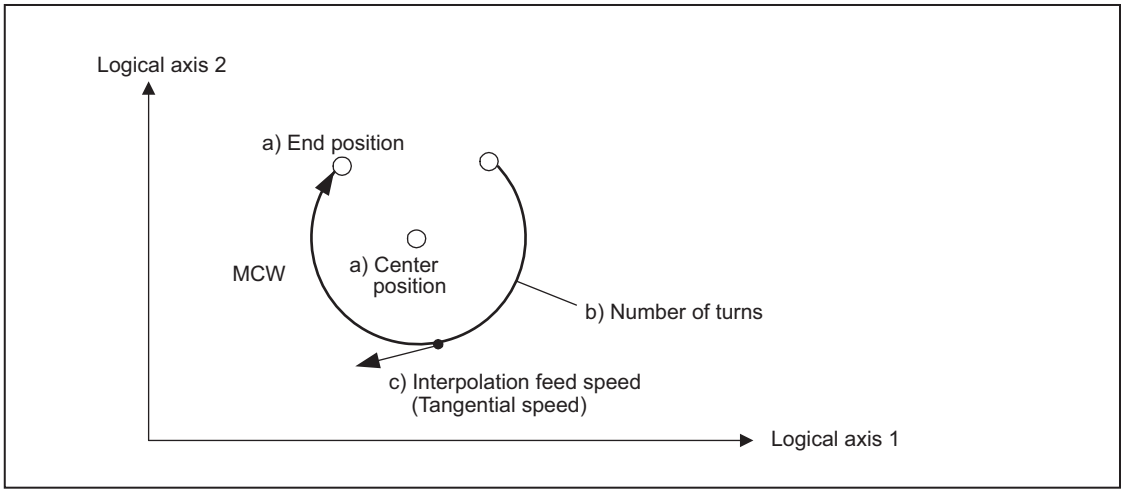
MCW [*Logical axis name 1*] *End position* [*Logical axis name 2*] *End position* *U Center position* *V Center position*
T Number of turns *F Interpolation feed speed* ;

Item	Unit	Usable Data
End position	Reference unit	・Directly designated value
Center point position	Reference unit	・Double integer register (Indirect designation)
Number of turns	Number of times of turn	
Interpolation feed speed	Reference units/min	

Note: The number of turns and interpolation feed speed can be omitted.

(3) Setting Items for MCW and MCC Commands

Motion Image

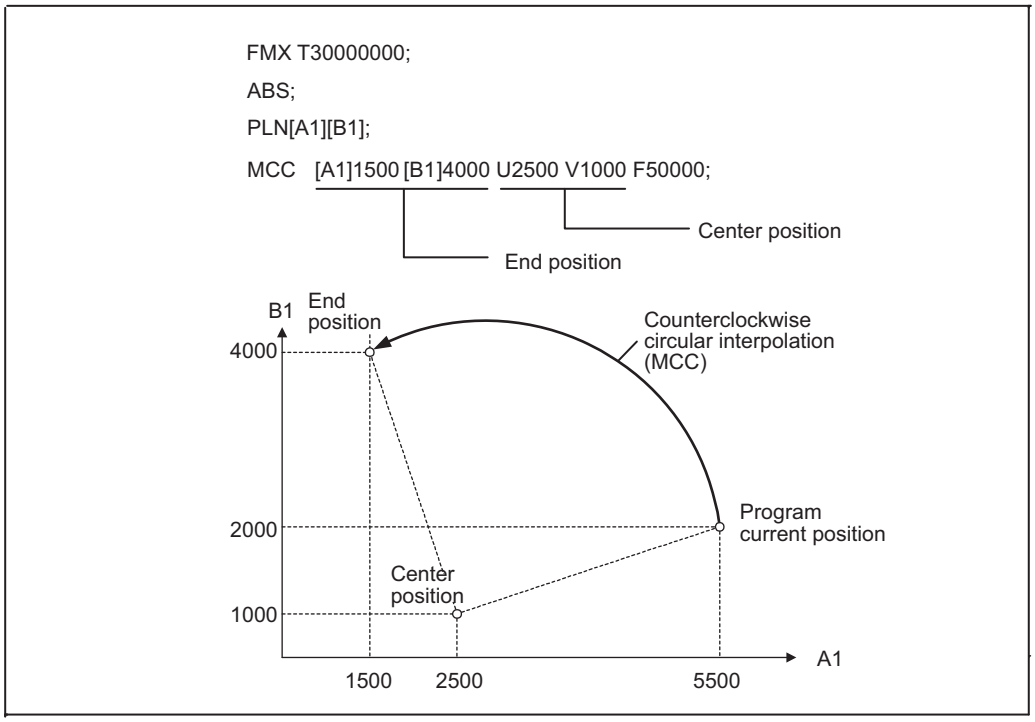


a) End position and Center point position

End position: Specify a numerical value or register following to a logical axis name.
Center position: Specify a numerical value or register following to character U and V in the MCW or MCC command block.
The actual end position and center position for the reference positions will differ, depending on the movement mode: ABS mode or INC mode.

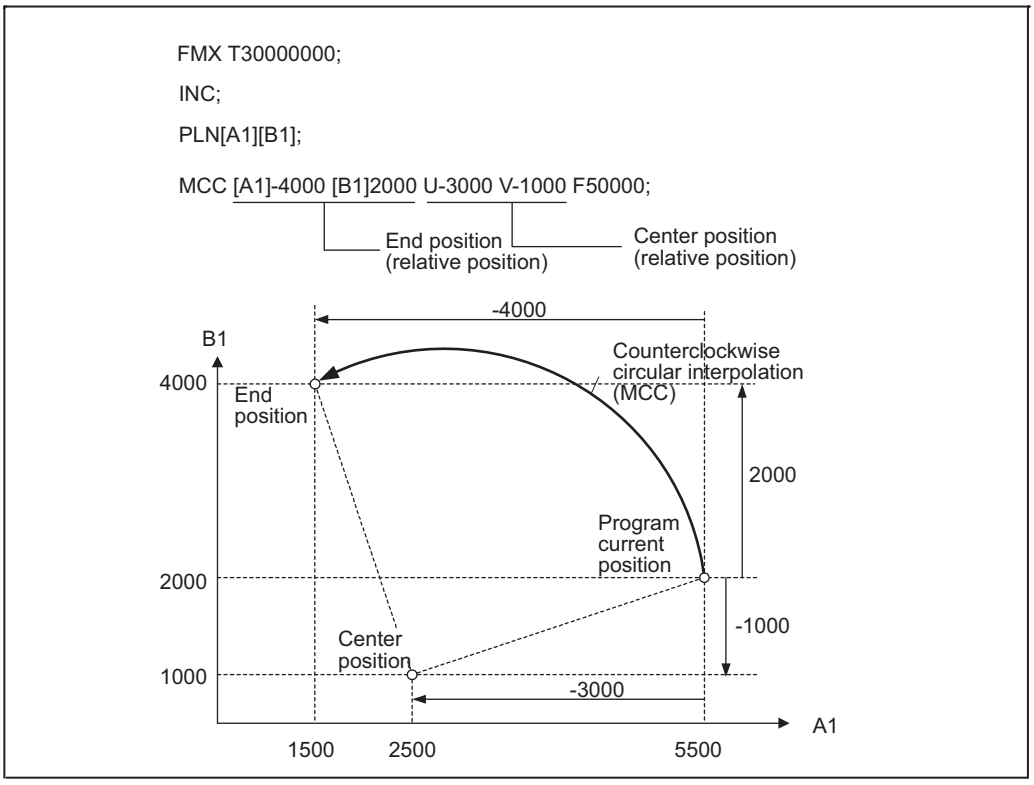
• In ABS Mode

The center position and end position are treated as absolute positions.



• In INC Mode

The center position and end position are treated as relative positions from the program current position.



IMPORTANT

Special care must be taken with regard to the start point radius, and end point radius, because the circular interpolation path will become as shown below if the start point radius is not equal to the end point radius.

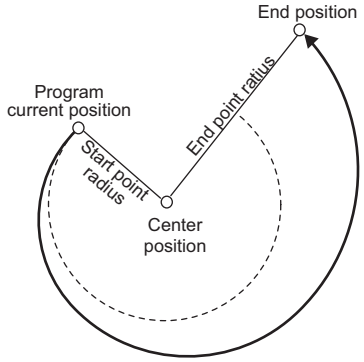


Fig. 8.30 Circular Interpolation Path When Start Point Radius ≠ End Point Radius

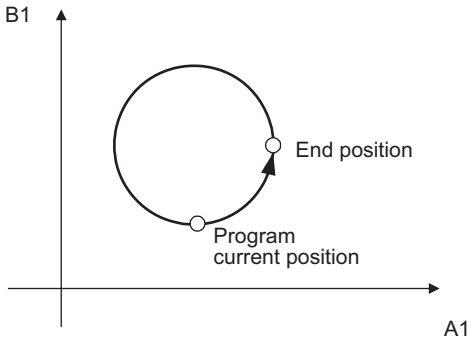
b) Number of turns

Specify a numerical value or register following to character T in the MCW or MCC command block. Specifying the number of turns will implement multiple circular movements. Specifying a negative value for the number of turns will cause a motion program alarm. The number of circular movements determined by the specified number of turns will differ depending on the relationship between the program current position and end position as shown below.

◀ EXAMPLE ▶

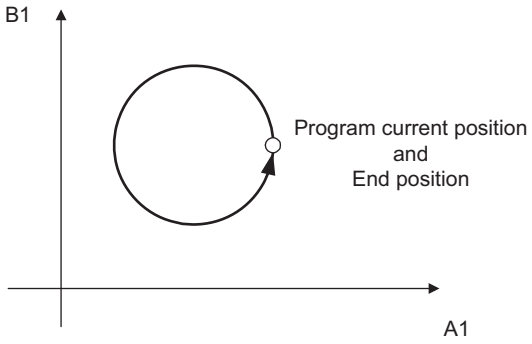
• When the number of turns is set to 2

When Program current position ≠ End position



Two circles + 1/4 circle

Program current position = End position



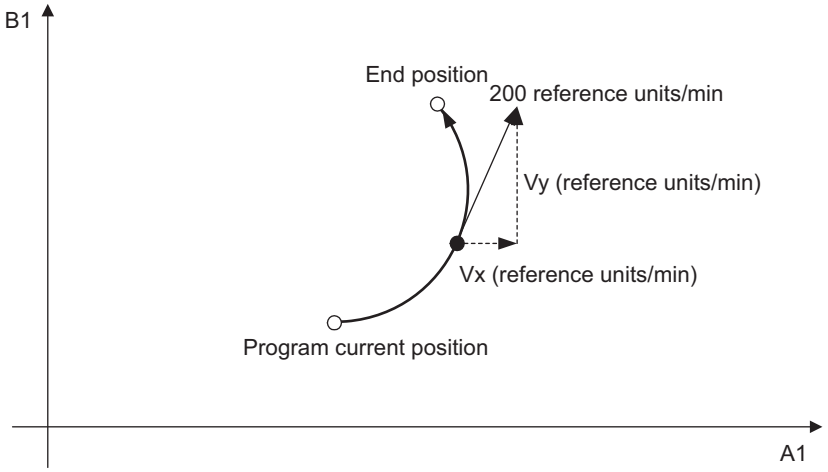
Three circles

c) Interpolation feed speed

For circular interpolation (MCW, MCC), the specified interpolation feed speed is treated as the speed of a tangential direction.
The reference range is between 1 and the maximum interpolation feed speed (FMX) [reference unit/min].

MCC command block: MCC[A1]- [B1]- F200;

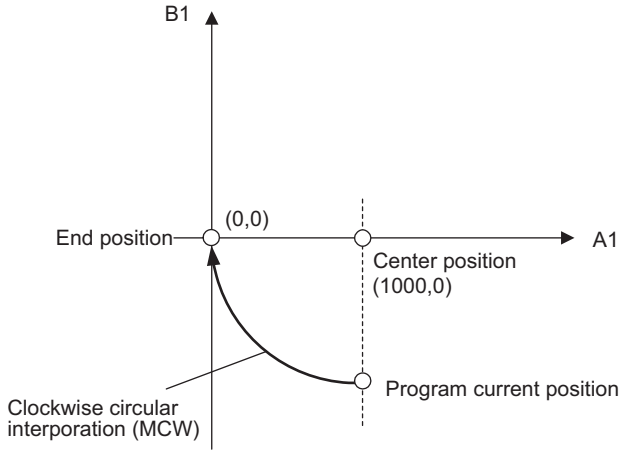
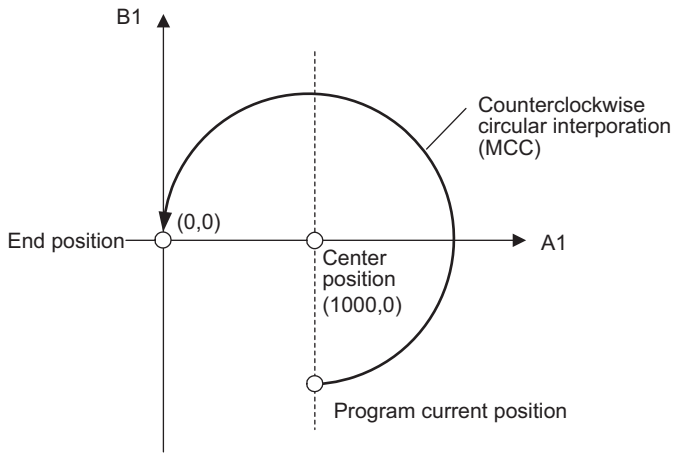
$F = 200 = \sqrt{Vx^2 + Vy^2}$ (reference units/min)



(4) Programming Examples

◀ EXAMPLE ▶

A programming example of circular interpolation (MCW, MCC) in ABS mode is shown below. The MCW commands turns axes clockwise, while MCC commands counterclockwise.

Turning Direction	Programming Example
Clockwise (MCW)	ABS; FMX T30000000; PLN [A1][B1]; MCW [A1]0 [B1]0 U1000 V0 F2000; “MCW (clockwise)” END;  Fig. 8.31 Center Position Designated Clockwise Circular Interpolation (MCW)
Counter-clockwise (MCC)	ABS; FMX T30000000; PLN [A1][B1]; MCC [A1]0 [B1]0 U1000 V0 F2000; “MCC (Counterclockwise)” END;  Fig. 8.32 Center Position Designated Counterclockwise Circular Interpolation (MCC)

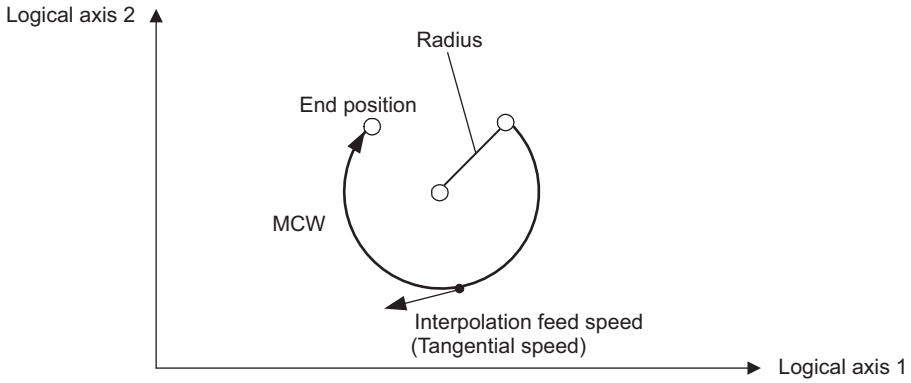
8.2.4 Clockwise/Counterclockwise Circular Interpolation with Radius Designation (MCW, MCC)

Motion Programs	Sequence Programs
Applicable	Not applicable

(1) Overview

The Clockwise/Counterclockwise Circular Interpolation with Radius Designation command (MCW, MCC) moves two axes simultaneously from the program current position to the end position on the designated plane at the interpolation speed on the circle determined by the radius.

- MCW: Circular Interpolation command for Clockwise (CW)
- MCC: Circular Interpolation command for Counterclockwise (CCW)



IMPORTANT

- Before designating a Clockwise/Counterclockwise Circular Interpolation command, be sure to designate the circular interpolation plane in Coordinate Plane Setting (PLN).
  If the PLN command is not coded, a motion program alarm will occur at execution of the MCW or MCC command.
- Designate the axes for the end position and center point position in the same order as the axes are specified in the PLN command.
- Be sure to code the FMX command at the beginning of the program to specify the maximum feed speed for interpolation.
  Otherwise, a motion program alarm will occur at execution of the MCW or MCC command.
- When the acceleration/deceleration time is not specified, the default acceleration/deceleration time of 0 ms will be applied.

INFO

For axis movement with the MCW or MCC command, an in-position check is not automatically executed. Use the PFN command to execute an in-position check if required.

(2) Format

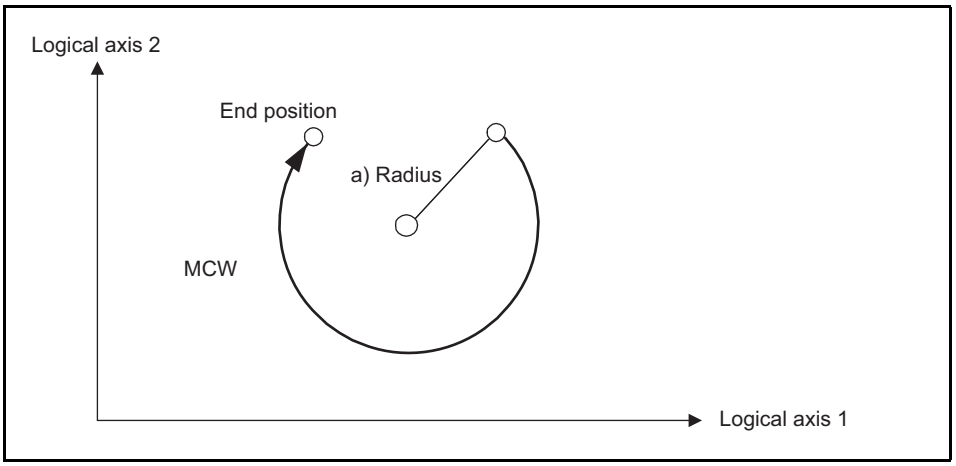
MCW [*Logical axis name 1*] *End position* [*Logical axis name 2*] *End position* *R Radius* *F Interpolation speed* ;

Item	Unit	Usable Data
End position	Reference unit	· Directly designated value · Double integer type register (Indirect designation)
Radius	Reference unit	
Interpolation feed speed	Reference units/min	

Notes: 1. The interpolation feed speed can be omitted.
2. With radius designating circular interpolation, the number of turns cannot be specified.

(3) Setting Items for MCW and MCC Commands

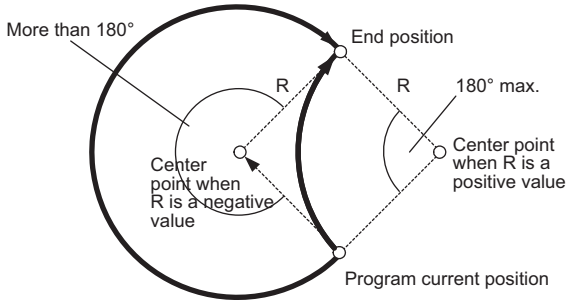
Motion Image



a) Radius

Specify a numerical value or register following to character R in the MCW or MCC command block.
The circular interpolation path will differ depending on the radius reference value as follows.

With the MCW command block MCW [A1] – [B1] – R –;
If R > 0: Circular interpolation with an arc angle of 180° or less
If R < 0: Circular interpolation with an arc angle of more than 180°
If R = 0: A motion program alarm will occur.



INFO With the radius designating circular interpolation, the number of turns cannot be specified.

(4) Programming Examples

◀ EXAMPLE ▶

Programming examples of the circular interpolation command (MCW, MCC) in ABS mode are shown below. The turning direction is specified by MCW (clockwise) and MCC (counterclockwise), and the arc angle is specified by the radius reference value.

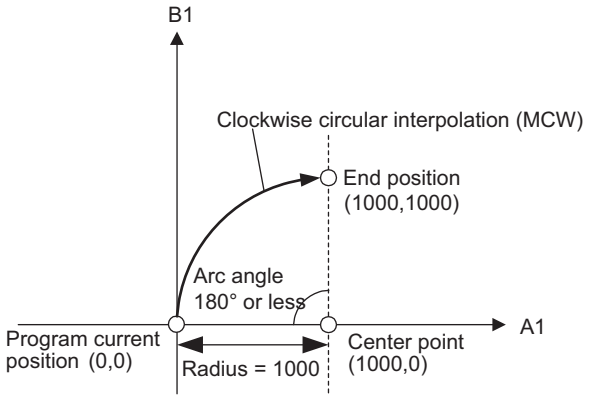
<table>
<tr><th>Turning Direction</th><th>Arc Angle</th><th>Programming Example</th></tr>
<tr><td rowspan="2">Clockwise (MCW)</td><td>180° or less
(Radius reference value > 0)</td><td>ABS;
FMX T30000000;
PLN [A1][B1];
MCW [A1]1000 [B1]1000 R1000 F2000; “MCW (Clockwise)”
END;
 

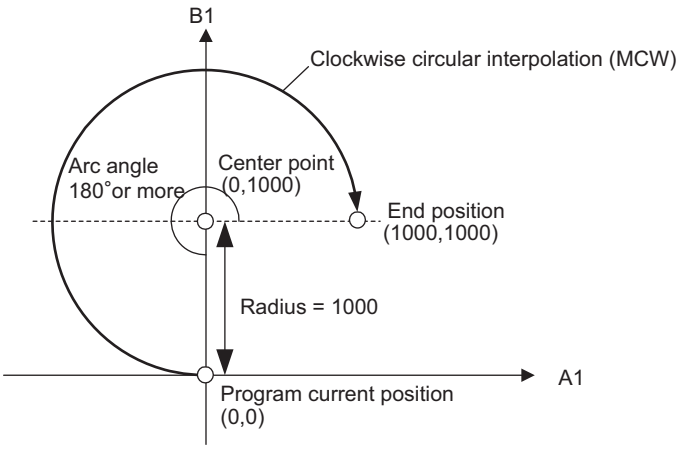
Fig. 8.33 Radius Designating Clockwise Circular Interpolation Command (MCW)</td></tr>
<tr><td>180° or more
(Radius reference value < 0)</td><td>ABS;
FMX T30000000;
PLN [A1][B1];
MCW [A1]1000 [B1]1000 R-1000 F2000; “MCW (Clockwise)”
END;
 

Fig. 8.34 Radius Designating Clockwise Circular Interpolation Command (MCW)</td></tr>
</table>

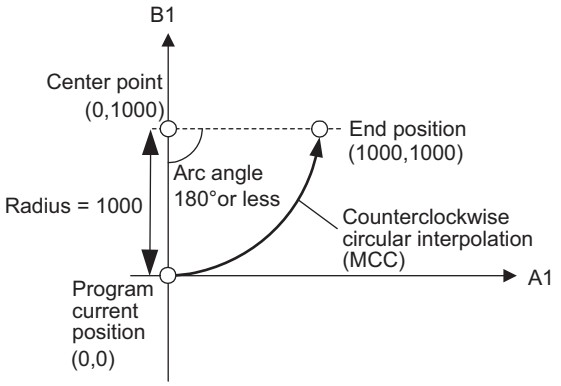
<table>
<tr><th>Turning Direction</th><th>Arc Angle</th><th>Programming Example</th></tr>
<tr><td rowspan="2">Counter-clockwise (MCC)</td><td>180° or less
(Radius reference value > 0)</td><td>ABS;
FMX T30000000;
PLN [A1][B1];
MCC [A1]1000 [B1]1000 R1000 F2000; “MCC (Counterclockwise)”
END;
 

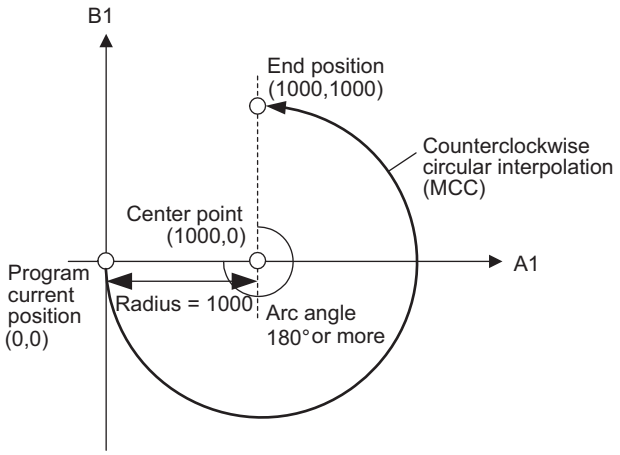
Fig. 8.35 Radius Designating Counterclockwise Circular Interpolation Command (MCC)</td></tr>
<tr><td>180° or more
(Radius reference value < 0)</td><td>ABS;
FMX T30000000;
PLN [A1][B1];
MCC [A1]1000 [B1]1000 R-1000 F2000; “MCC (Counterclockwise)”
END;
 

Fig. 8.36 Radius Designating Counterclockwise Circular Interpolation Command (MCC)</td></tr>
</table>

8.2.5 Clockwise/Counterclockwise Helical Interpolation with Center Position Designation (MCW, MCC)

Motion Programs	Sequence Programs
Applicable	Not applicable

(1) Overview

The Clockwise/Counterclockwise Helical Interpolation with Center Position Designation command (MCW, MCC) simultaneously executes a linear interpolation movement while moving on the circle (circular interpolation) determined by the designated center point position.
The helical interpolation feed speed is calculated by using the tangential speed for circular interpolation and composite speed for linear interpolation.

- MCW: Helical Interpolation command for Clockwise (CW)
- MCC: Helical Interpolation command for Counterclockwise (CCW)

⚠ CAUTION

- The linear interpolation axis specified for a Clockwise/Counterclockwise Helical Interpolation (MCW, MCC) command can be either a linear axis or a rotary axis. Depending on the axis movement in the linear interpolation portion, the helical interpolation path may not be a helical shape. When programming, be sure to check the path to make sure that there are no tools or other obstacles in the way of the workpiece.
  Failure to carry out this check may result in damage to equipment, serious personal injury, or even death.

IMPORTANT

- Be sure to specify the plane for circular interpolation by using the Coordinate Plane Setting command (PLN) before executing the Helical Interpolation command (MCW or MCC).
  Use logical axis 1 and logical axis 2 to specify the end positions and center points of circle of the horizontal and vertical axes of the designated plane.
- Specify the axes for the end position and center position in the same order as the axes are specified in the PLN command.
- Any axis that has not been specified in the plane designation can be specified as a linear interpolation axis. The axis does not need to be at right angles to the interpolation plane.

INFO

For an axis movement initiated by execution of the helical interpolation command MCW or MCC, an in-position check to check whether the axis enters the positioning completion range will not be executed.
Use the PFN command to execute an in-position check if required.

(2) Format

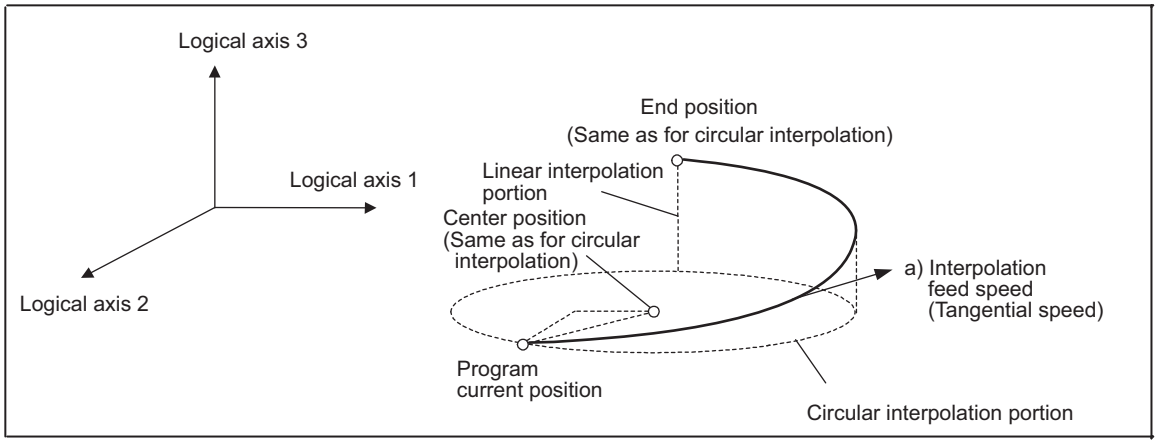
MCW [*Logical axis name 1*] *End position* [*Logical axis name 2*] *End position* *U Center point position* *V Center point position*
[*Logical axis name 3*] *End position for linear interpolation* *T Number of turns* *F Interporation feed speed* ;

Item	Unit	Usable Data
End position	Reference unit	· Directly designated value · Double integer type register (Indirect designation)
Center point position	Reference unit	
Number of turns	Number of times of turn	
Interpolation feed speed	Reference unit/min	

Note: The number of turns and interpolation feed speed can be omitted.

(3) Setting Items for MCW and MCC Commands

Motion Image

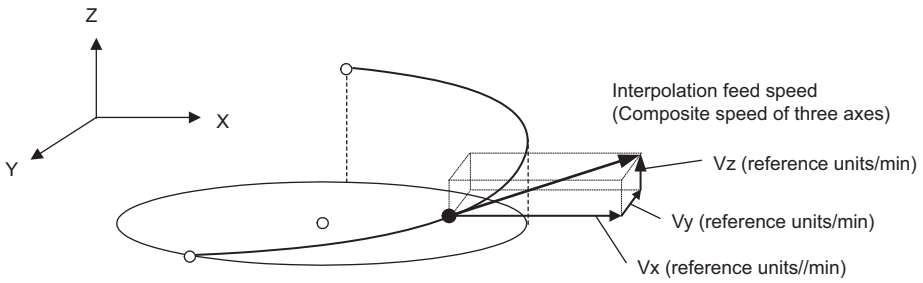


a) Interpolation feed speed

For the helical interpolation commands (MCW and MCC), the interpolation feed speed is calculated with the tangential speed for circular interpolation and the composite speed for the linear interpolation axis.

MCC command block: *MCC [X]- [Y]- U- V- [Z]- F300;*

$F = 300 = \sqrt{Vx^2 + Vy^2 + Vz^2}$ (reference units/min)



(4) Programming Examples

◀ EXAMPLE ▶

A programming example of a clockwise helical interpolation command (MCC) in ABS mode is shown below.

```
ABS;
FMX T30000000;
PLN [A1][B1];
MCC [A1]1000 [B1]0 U0 V0 [C1]500 F2000;
END;
```

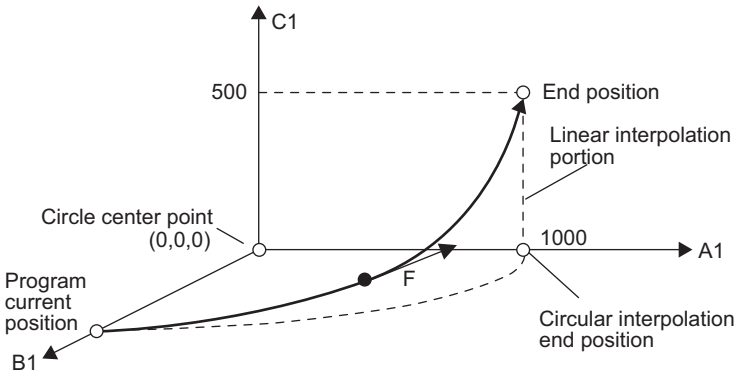


Fig. 8.37 Center Point Position Designating Clockwise Helical Interpolation Command (MCC) Programming Example

8.2.6 Clockwise/Counterclockwise Helical Interpolation with Radius Designation (MCW, MCC)

Motion Programs	Sequence Programs
Applicable	Not applicable

(1) Overview

The Clockwise/Counterclockwise Helical Interpolation with Radius Designation command (MCW, MCC) simultaneously executes a linear interpolation movement while moving on the circle (circular interpolation) determined by the designated radius.
The helical interpolation feed speed is calculated by using the tangential speed for circular interpolation and the composite speed for linear interpolation.

- MCW: Helical Interpolation command for Clockwise (CW)
- MCC: Helical Interpolation command for Counterclockwise (CCW)

⚠ CAUTION

- The linear interpolation axis specified for a Clockwise/Counterclockwise Helical Interpolation (MCW, MCC) command can be either a linear axis or a rotary axis. Depending on the axis movement in the linear interpolation portion, the helical interpolation path may not be a helical shape. When programming, be sure to check the path to make sure that there are no tools or other obstacles in the way of the workpiece.
  Failure to carry out this check may result in damage to equipment, serious personal injury, or even death.

IMPORTANT

- Be sure to specify the plane for circular interpolation by using the Coordinate Plane Setting command (PLN) before executing the Helical Interpolation command (MCW or MCC).
  Use logical axis 1 and logical axis 2 to specify the end positions and center points of circle of the horizontal and vertical axes of the designated plane.
- Specify the axes for the end position and center position in the same order as the axes are specified in the PLN command.
- Any axis that has not been specified in the plane designation can be specified as a linear interpolation axis. The axis does not need to be at right angles to the interpolation plane.

INFO

For an axis movement initiated by execution of the helical interpolation command MCW or MCC, an in-position check to check whether the axis enters the positioning completion range will not be executed.
Use the PFN command to execute an in-position check if required.

(2) Format

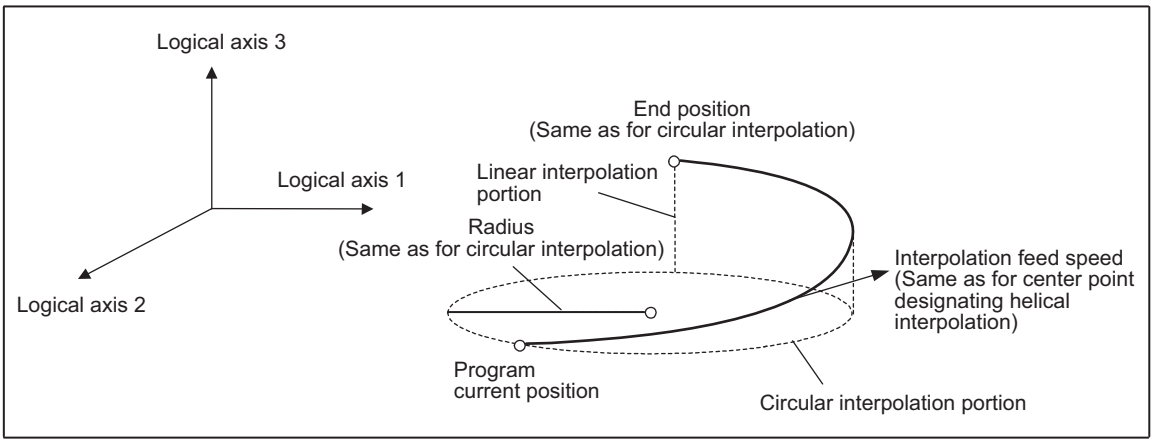
MCW [*Logical axis name 1*] *End position* [*Logical axis name 2*] *End position* *R Radius*
[*Logical axis name 3*] *End position for linear interpolation* *F Interpolation feed speed* ;

Item	Unit	Usable Data
End position	Reference unit	· Directly designated value · Double integer type register (Indirect designation)
Center point position	Reference unit	
Radius	Reference unit	
Interpolation feed speed	Reference units/min	

Notes: 1. The interpolation feed speed can be omitted.
2. With the radius designating helical interpolation command, the number of turns cannot be designated.

(3) Setting Items for MCW and MCC Commands

Motion Image



INFO

The designation methods of the radius and end position for the radius designating helical interpolation command are the same as for the radius designating circular interpolation command.

Additionally, the designation method of the interpolation feed speed is the same as for the center position designating helical interpolation command.

(4) Programming Examples

◀ EXAMPLE ▶

A programming example of a radius designating counterclockwise helical interpolation command (MCC) is shown below.

```
ABS;
FMX T30000000;
PLN [A1][B1];
MCC [A1]1000 [B1]0 R1000 [C1]500 F2000;
END;
```

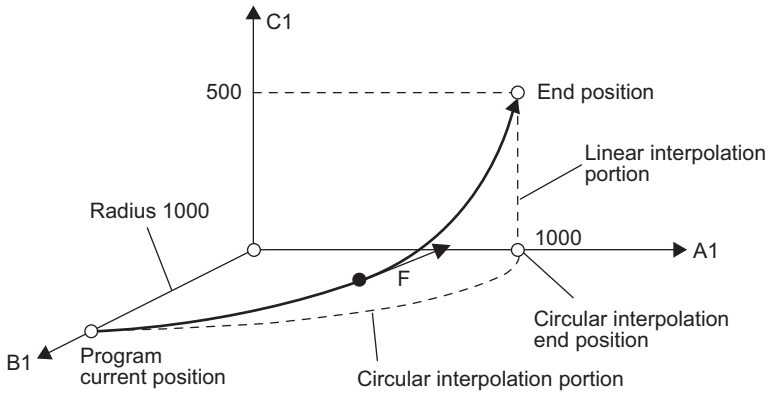


Fig. 8.38 Radius Designating Counterclockwise Helical Interpolation Command (MCC)

8.2.7 Zero Point Return (ZRN)

Motion Programs	Sequence Programs
Available	Not Available

(1) Overview

The Zero Point Return (ZRN) command executes the zero point return operation. Up to 16 axes can be designated simultaneously. An axis unspecified in the command block will not move. The resulting stop position is set as the machine coordinate origin.

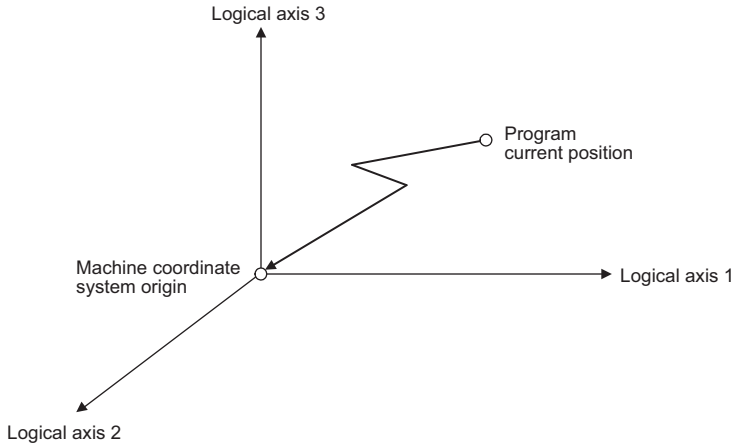


Fig. 8.39 Movement Path with ZRN Command

When the ZRN command is executed, the returned position is set as the machine coordinate origin. At the same time, the work coordinate system previously set by Current Position Set (POS) is cancelled.

After the ZRN command has been executed, the machine coordinate system will be the same as the work coordinate system. Until the next time that Current Position Set (POS) is executed, Move ON Machine Coordinates (MVM) will be ineffective, even if it is designated.

Refer to *8.3.1 Current Position Set (POS)* for details of the machine coordinate system and work coordinate system.

IMPORTANT

Request for temporary stop of program is disabled during ZRN command execution. To stop the operation on the way, execute request for stop of program.

Refer to *4.3.3 Work Registers* for information on Program Pause Request and Program Stop Request signals.

(2) Format

ZRN [*Logical axis name 1*] 0 [*Logical axis name 2*] 0 [*Logical axis name 3*] 0 ... ;

Note: Always code “0” after each logical axis name.

(3) Setting Items for ZRN Command

(a) Zero Point Return Methods

Use motion setting parameter OW□□3C (Zero point return method) to set the zero point return method for each axis.
The table below lists the available zero point return methods.
Refer to the relevant motion module manual for details on each method.

Zero Point Return Method	Setting (OW□□3C)	SVA-01	SVB-01/ Built-in SVB	PO-01
DEC1 + Phase-C pulse	0	Available	Available	N/A
ZERO signal	1	Available	Available	N/A
DEC1 + ZERO signal	2	Available	Available	Available
Phase-C pulse	3	Available	Available	N/A
DEC2 + ZERO signal	4	Available	N/A	Available
DEC1 + LMT + ZERO signal	5	Available	N/A	Available
DEC2 + Phase-C signal	6	Available	N/A	N/A
DEC1 + LMT + Phase-C signal	7	Available	N/A	N/A
C pulse only	11	Available	Available	N/A
P-OT & Phase-C pulse	12	Available	Available	N/A
P-OT	13	Available	Available	N/A
HOME LS & Phase-C pulse	14	Available	Available	N/A
HOME LS	15	Available	Available	N/A
N-OT & Phase-C pulse	16	Available	Available	N/A
N-OT	17	Available	Available	N/A
INPUT & Phase-C pulse	18	Available	Available	N/A
INPUT	19	Available	Available	N/A

(b) Zero Point Return Speed

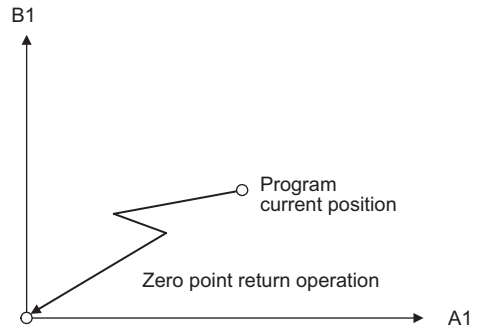
The zero point return speed depends on the applied zero point return method.
For details, refer to the relevant motion module manual.

(4) Programming Examples

◀ EXAMPLE ▶

A programming example of a ZRN command in ABS mode is shown below.

```
ZRN [A1]0 [B1]0;
END;
```



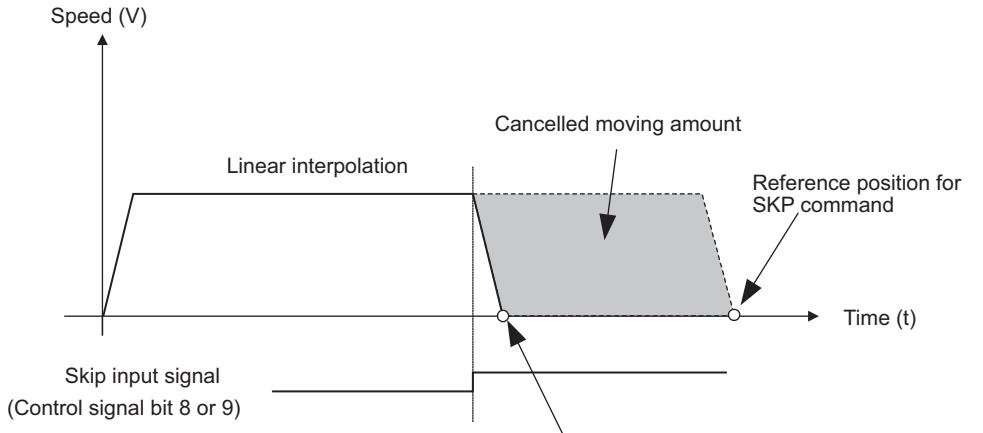
The stop position is set to the machine coordinate origin (0, 0).

8.2.8 Linear Interpolation with Skip Function (SKP)

Motion Programs	Sequence Programs
Applicable	Not applicable

(1) Overview

The Linear Interpolation with Skip Function command (SKP) is an extended command of the Linear Interpolation command (MVS). When the skip input signal is turned ON during axis movement by a SKP command, the moving axis is decelerated to a stop and the remaining travel distance is cancelled.
A use of a SKP command enables the programming of motion control that can respond to external conditions.
The skip signal is input to the control signal for the MSEE command or the control register of M-EXECUTOR.



Skip Input Signal Selection	Skip Input Signal
Skip input signal 1 (SS1)	Bit 8 of control signal
Skip input signal 2 (SS2)	Bit 9 of control signal

Fig. 8.40 Axis Movement by SKP Command

IMPORTANT

The moving axis decelerates to a stop when the skip input signal is turned ON. The SKP command, however, remains in effect until the positioning completion signal is turned ON.

(2) Format

SKP [*Logical axis name 1*] *Reference position* [*Logical axis name 2*] *Reference position* [*Logical axis name 3*] *Reference position*...
F Interporation feed speed *SS Skip input signal selection* ;

Item	Unit	Usable Data
Reference position	Reference unit	· Directly designated value · Double interger type register (Indirect designation)
Interpolation feed speed	Reference unit/min	
Skip input signal selection	—	· Directly designated number 1 or 2 · Double integer type register (Indirect designation)

Note: The interpolation feed speed can be omitted.

(3) Programming Examples

◀ EXAMPLE ▶

A programming example of a SKP command in ABS mode is shown below.

```
FMX T30000000;
ABS;
IAC T1000;
IDC T1000;
SKP [A1]4000 [B1]3000 [C1]2000 F50000 SS1;
END;
```

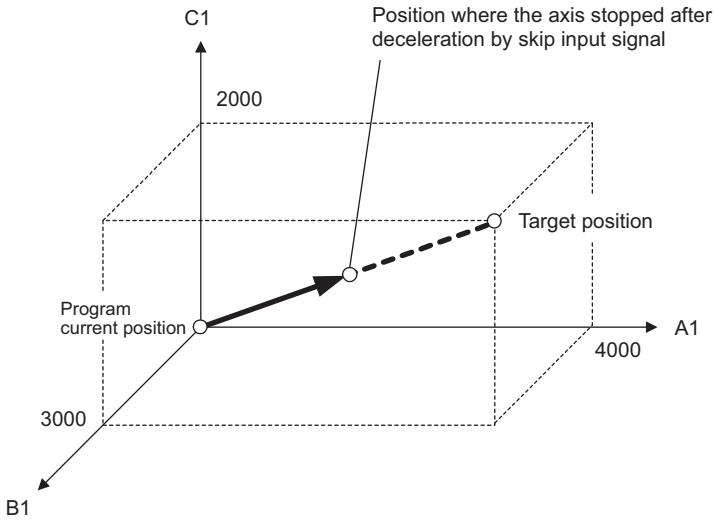


Fig. 8.41 SKP Command Programming Example

8.2.9 Set Time Positioning (MVT)

Motion Programs	Sequence Programs
Applicable	Not applicable

(1) Overview

The Set Time Positioning command (MVT) is an extended Positioning command (MOV).
The MVT command can simultaneously move up to 16 axes. An axis unspecified in the MVT command block will not move.
With the MVT command, the feed speed of each axis is adjusted to complete positioning in the specified time.
The MVT command does not use an interpolation operation, and there is no restriction on completing the positioning for all the specified axes simultaneously.
There is a time lag caused by the acceleration/deceleration setting.

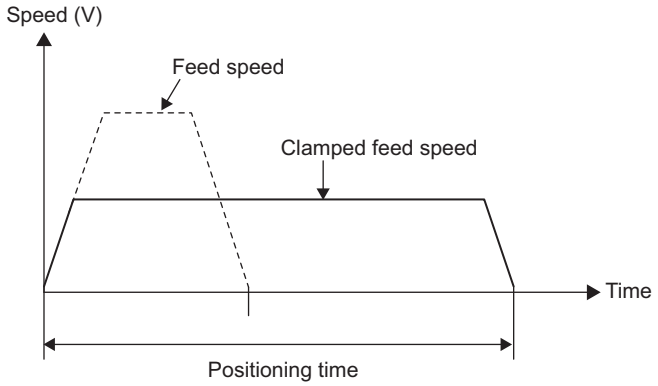


Fig. 8.42 Axis Movement with MVT Command

If an override is used, positioning will not complete in the specified time.
If a filter is used, the positioning time will be delayed by the filter time constant.

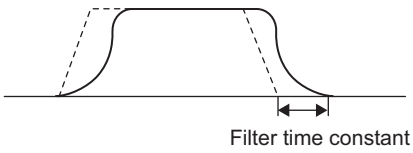


Fig. 8.43 Positioning Time Delay When a Filter is used

IMPORTANT

- If the positioning time is set to 0, an alarm will occur in the motion program.
- If the moving amount of any of the specified axes is set to 0, an alarm will occur in the motion program.

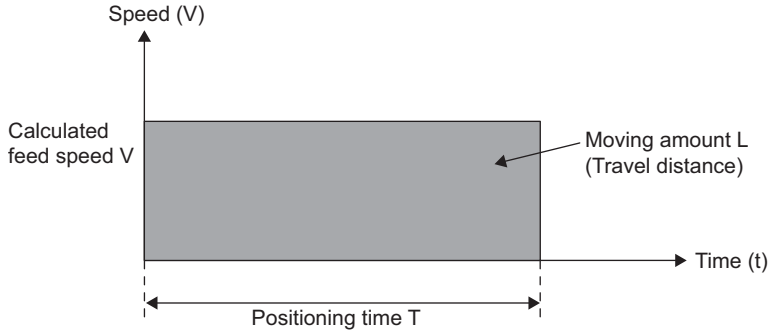
(2) Format

MVT [*Logical axis name 1*] *Reference position* [*Logical axis name 2*] *Reference position* [*Logical axis name 3*] *Reference position* ...

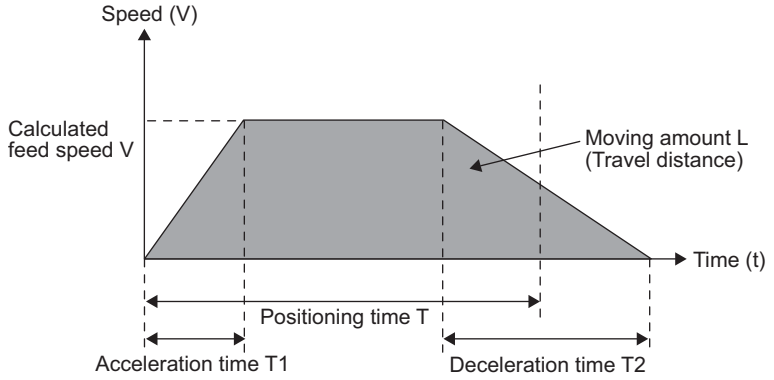
T Positioning time ;

Item	Unit	Usable Data
Position reference	Reference unit	· Directly designated value
Positioning time	ms	· Double integer type register (Indirect designation)

The positioning time reference range is between 1 and 2147483647 (ms).
Inside the MP2000 Machine Controller, the feed speed during MVT command execution is calculated according to the specified positioning time and moving amount.
This calculation does not include acceleration (acceleration = 0) as shown below.



The actual operation when the acceleration time T1 is less than the deceleration time T2 will be as shown below.



The feed speed set by VEL command will be changed accordingly. After executing the MVT command, reset the feed speed by using VEL command.

INFO

An in-position check is executed for axis movement by MVT command execution, as well as for MOV command execution, to verify that the axis enters the positioning completed range.

(3) Programming Examples

◀ EXAMPLE ▶

A programming example of an MVT command in ABS mode is shown below.

```
ABS;
ACC [A1]1000;
DCC [A1]1000;
MVT [A1]4000 T1000;
END;
```

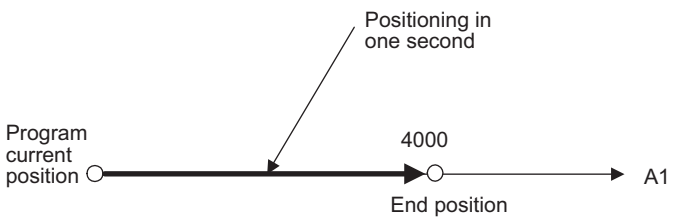


Fig. 8.44 MVT Command Programming Example

8.2.10 External Positioning (EXM)

Motion Programs	Sequence Programs
Applicable	Not applicable

(1) Overview

The External Positioning command (EXM) is an extended Positioning command (MOV).
The EXM command executes positioning by using an incremental value to move the axis the specified moving amount when the external positioning signal is turned ON. If the external positioning signal is not turned ON, the axis completes positioning at the reference position specified in the EXM command block.
Only one axis can be specified in an EXM command block.

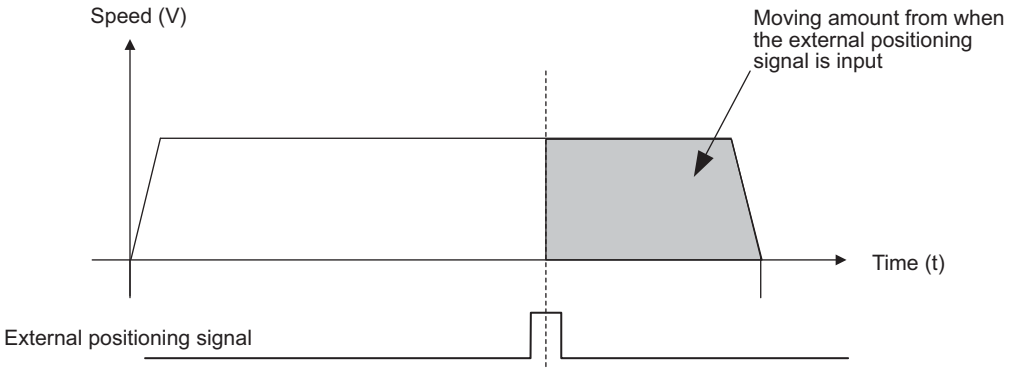


Fig. 8.45 Movement by EXM Command

When a negative value is specified for the moving amount, the axis decelerates to a stop and then moves in the negative direction.

IMPORTANT

- For the external positioning signal, refer to the relevant motion module manual.
- The External Positioning command (EXM) cannot be used with a PO-01 module.
  If used, an alarm will occur in the motion program.
- Special care must be taken to use the external latch input signal, because it is also used for the zero point return operation.

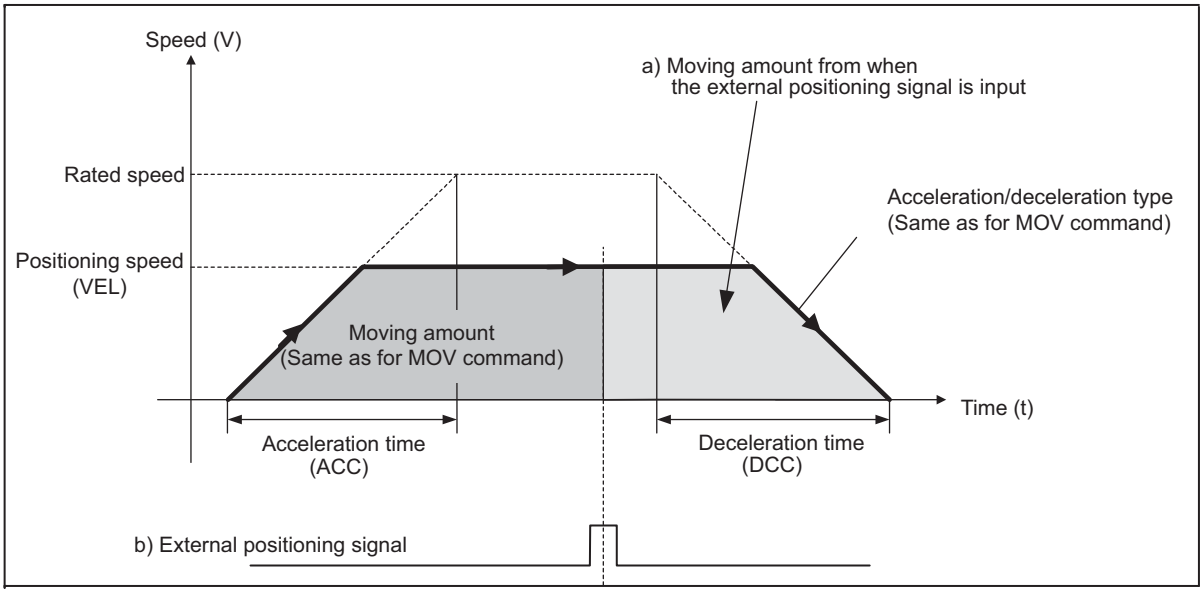
(2) Format

EXM [*Logical axis name 1*] *Reference position* *D Moving amount from when the external positioning signal is input* ;

Item	Unit	Usable Data
Reference position	Reference unit	・Directly designated value ・Double integer type register (Indirect designation)
Moving amount from when the external positioning signal is input	Reference unit	

(3) Setting Items for EXM Command

Motion Image



a) Moving amount from when the external positioning signal is input

Set the moving amount using an incremental value after the external positioning signal is turned ON.
The reference range is between -2147483648 to +2147483647 (reference unit).

b) External positioning signal

Use bits 4 to 7 (External positioning signal setting) of motion setting parameter OW□□04 (Function setting 2) to select the external positioning signal.
For details, refer to the relevant motion module manual.

INFO The PO-01 module does not have the external positioning function.

(4) Programming Examples

◀ EXAMPLE ▶

A programming example of EXM command in ABS mode is shown below.

```
ABS;
ACC [A1]1000;
DCC [A1]1000;
VEL [A1]2000;
DL00000 = 1000;
EXM [A1]4000 DDL00000;
END;
```

8.3 Axis Control Commands

This section describes the axis control commands.

8.3.1 Current Position Set (POS)

Motion Programs	Sequence Programs
Applicable	Not applicable

⚠ CAUTION

- Care is required with the Current Position Set (POS) command.
  The Current Position Set (POS) command is used to create new work coordinate system values. If POS is specified incorrectly, subsequent move operations will be entirely different. Before starting operations, be sure to check that the work coordinate system is specified correctly.
  Failure to carry out this check may result in damage to equipment, serious personal injury, or even death.

(1) Overview

The Current Position Set command (POS) changes the current position to a desired coordinate value to create a new coordinate system. In this manual, the newly set coordinate system is referred to as the "work coordinate system" while the original coordinate system of machine is referred to as the "machine coordinate system." Move commands coded after a POS command will be executed to move axes in the work coordinate system.

Coordinate System	Description	Remarks
Machine Coordinate System	Original coordinate system of machine	The position for zero point return is the origin (0).
Work Coordinate System	Coordinate system constructed by the user defined position	A new coordinate system constructed by POS command execution

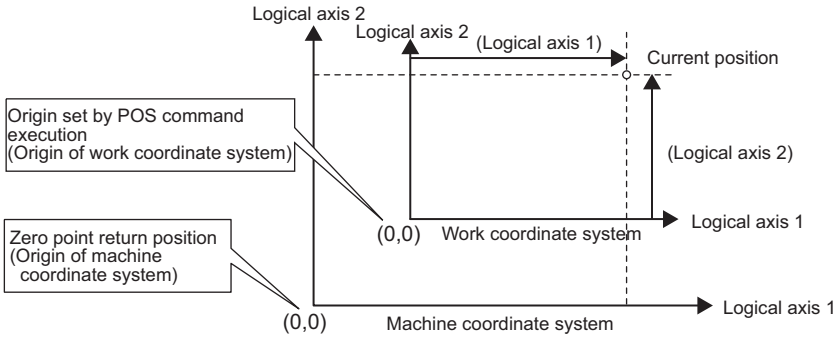


Fig. 8.46 Work Coordinate System Setting by POS Command

The work coordinate system can be changed as often as desired by using the POS command. The machine coordinate system must be set in advance. The machine coordinate system is not affected by the POS command.
Up to 16 axes can be designated in a POS command block. The work coordinate system for an axis not designated in the POS command block will not be constructed or updated.
Move commands in a work coordinate system cannot exceed the maximum programmable value when converted to coordinates in the machine coordinate system.

The following table shows the setting status of the machine coordinate system and the work coordinate system.

Table 8.1 Coordinate System Setting Timing

Coordinate System Setting Timing	Motion Fixed Parameter 30: Encoder Selection	
	0 or 2: Incremental Encoder/Absolute Encoder (Incremental Encoder is Used)	1: Absolute Encoder
After power ON	Machine coordinate system: Default setting *1 Work coordinate system: Cancelled *3	Machine coordinate system: Yes *2 Work coordinate system: Cancelled
After Zero Point Return command (ZRN)	Machine coordinate system: Set Work coordinate system: Cancelled	Work coordinate system: Cancelled
After POS command	Work coordinate system: Set	Work coordinate system: Set
After Zero Point Set (ZSET) command	Machine coordinate system: Set	Machine coordinate system: Set

* 1. Default setting: The current position is set as the machine coordinate origin when the power is turned ON. If the Zero Point Return operation is then not executed, the software limit switch function will not be effective.
* 2. Yes: The machine coordinate origin is set using the position information in the absolute position detection encoder.
* 3. Cancelled: The previously set work coordinate system is cancelled, and the work coordinate system is the same as the machine coordinate system.

IMPORTANT

- For infinite-length axes, set a value within the range of 0 to POSMAX.
  If a value outside the setting range is set for any infinite-length axis, an alarm will occur in the motion program.
- When the zero point return operation is executed without using a ZRN command, such as zero point return operation executed from the ladder program, the work coordinate system will not be cancelled.

(2) Format

POS [*Logical axis name 1*] *Coordinate axis* [*Logical axis name 2*] *Coordinate axis* ... ;

Item	Unit	Usable Data
Coordinate axis	Reference unit	· Directly designated value · Double integer type regisger (Indirect designation)

(3) Programming Examples

◀ EXAMPLE ▶

A POS command programming example is shown below.

```
ABS;                              " Absolute mode

MOV [A1]1000 [B1]2000;            " Positioning

POS [A1]0 [B1]0;                  " Set work coordinate system
MOV [A1]3000 [B1]4000;            " Positioning

DL00000 = IL8010;                 " Get Machine Coordinate System Calculated Position (CPOS) of Axis A1
DL00002 = IL8090;                 " Get Machine Coordinate System Calculated Position (CPOS of Axis B1
POS [A1]DL00000 [B1]DL00002;      " Cancel work coordinate system

END;
```

8.3.2 Move On Machine Coordinates (MVM)

Motion Programs	Sequence Programs
Applicable	Not applicable

⚠ CAUTION

- The Move ON Machine Coordinates (MVM) command is used to position the coordinate positions in a machine coordinate system. If the machine coordinate origin is designated without being verified, unexpected move operations will result. Before starting operations, be sure to check that the position designated in the machine coordinate system is correct.

  Failure to carry out this check may result in damage to equipment, serious personal injury, or even death.

(1) Overview

The Move ON Machine Coordinates (MVM) command is used to move axes in a machine coordinate system after a work coordinate system that is different from the machine coordinate system has been set by Current Position Set (POS).

Specifying an MVM command for an axis move command temporarily moves the axis to the absolute coordinate position in the machine coordinate system. During execution of an MVM command, the axis moves in ABS mode regardless of the movement mode setting.
The MVM command is valid only in the MVM command coded block. For example, the axis moves in the work coordinate system in the linear interpolation (MVS) in the next block or following blocks.

(2) Format

```
MVM  MOV ..... ;
or
MVM  MVS ..... ;
```

(3) Programming Examples

◀ EXAMPLE ▶

An MVM command programming example is shown below.

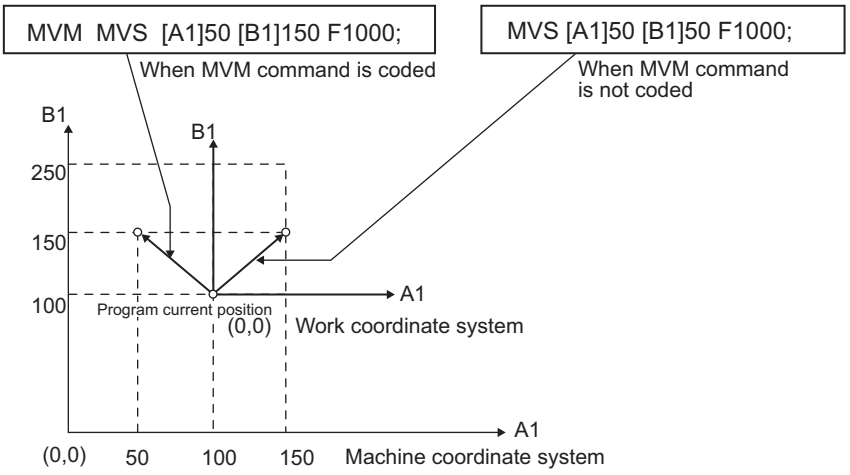


Fig. 8.47 Move On Machine Coordinates Command (MVM) Programming Example

8.3.3 Program Current Position Update (PLD)

Motion Programs	Sequence Programs
Applicable	Not applicable

(1) Overview

The Program Current Position Update command (PLD) updates a program current position that has been shifted by a manual intervention, etc.
If an axis movement is executed outside a motion program while the motion program is running (such as when an axis is moved by JOG, STEP, or a user function), the program current position will not be updated. If the motion program is executed in this status, the axis will move to the position shifted for the moving amount by manual intervention. To solve this problem, a PLD command is used to update the program current position.

(2) Format

PLD [*Logical axis name 1*] [*Logical axis name 2*] [*Logical axis name 3*] ... ;

(3) Programming Examples

PLD programming examples are shown below.

(a) Manual intervention while the motion program is running

◀ EXAMPLE ▶

```
MOV [A1]1000;
                  " Axis [A1] is moved by JOG during this command block is being executed.
PLD [A1];         " Updates the program current position.
MOV [A1]2000;
```

(b) Axis is moved within the motion program user function

◀ EXAMPLE ▶

```
MOV [A1]1000;
UFC  FNC10 MB000000 IW0100 MB000020;   "  Axis [A1] is moved by a user function
PLD [A1];                              "  Updates the program current position.
MOV [A1]2000;
```

INFO

The PLD command can be executed by the user in some applications. The PLD command cannot be used in some of the applications where manual intervention is required while the motion program is running.

8.3.4 In-Position Check (PFN)

Motion Programs	Sequence Programs
Applicable	Not applicable

(1) Overview

The In-Position Check command (PFN) verifies whether the axis being moved by an interpolation related command enters the NEAR position range (NEAR signal output width).
For an axis that is being moved by interpolation related command MVS, MCW, MCC, or SKP, an in-position check is normally not executed. Use a PFN command to verify that the axis enters the NEAR position range.

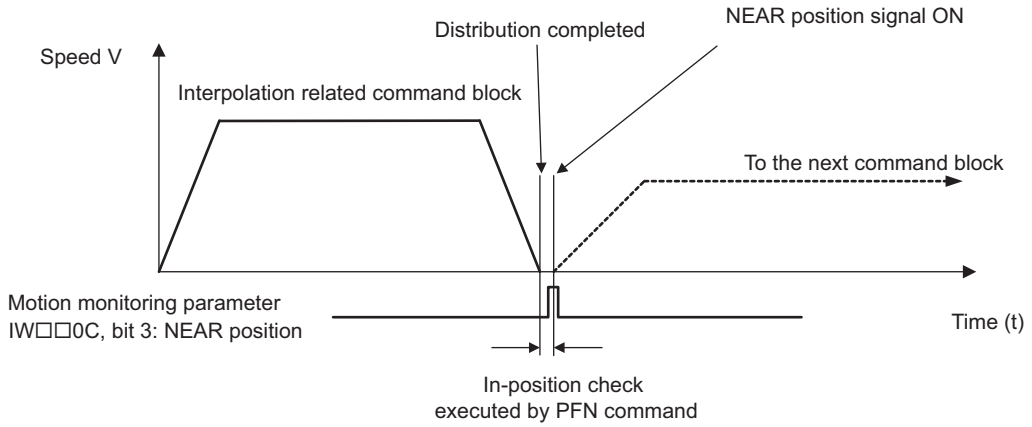


Fig. 8.48 In-position Check Executed by PFN Command

Bit 3 (NEAR position) of motion monitoring parameter IW□□0C turns ON when the following condition is satisfied.
Use INP command to set the NEAR signal output width.

$|$ MPOS – APOS $|$ ≤ NEAR signal output width

MPOS : Motion monitoring parameter IL□□12 (Machine coordinate system reference position)
APOS : Motion monitoring parameter IL□□16 (Machine coordinate system feedback position)

INFO

If the NEAR signal output width is set to 0, bit 3 of motion monitoring parameter IW□□0C turns ON when the distribution of the reference value, including the filter, is completed.

(2) Format

- When coding a PFN command in the interpolation related command block
  MVS [*Logical axis name 1*] - [*Logical axis name 2*] - [*Logical axis name 3*] ... PFN;

- When coding a PFN command independently
  PFN [*Logical axis name 1*] [*Logical axis name 2*] [*Logical axis namd 3*]...;

(3) Programming Examples

PFN command programming examples are shown below.

(a) When coding a PFN command in the interpolation related command block

◀ EXAMPLE ▶

```
MVS [A1]1000 F20000 PFN;
MOV [A1]3000;
END;
```

(b) When coding a PFN command independently

◀ EXAMPLE ▶

```
MVS [A1]1000 F20000;
PFN [A1];
MOV [A1]3000;
END;
```

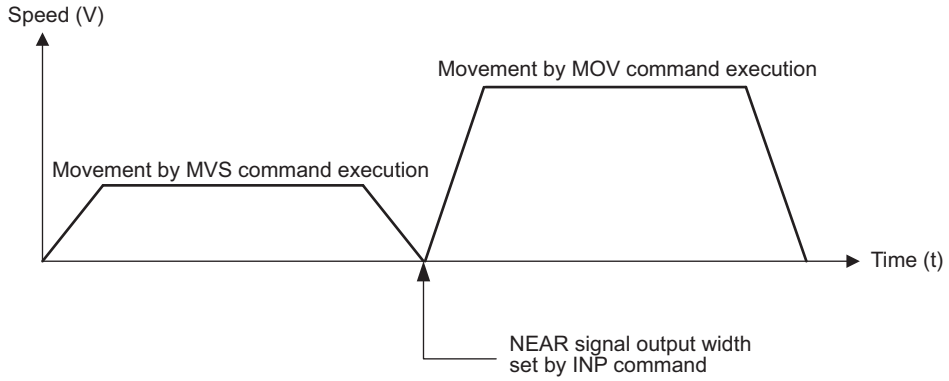


Fig. 8.49 PFN Command Programming Example

8.3.5 Set In-Position Range (INP)

Motion Programs	Sequence Programs
Applicable	Not applicable

(1) Overview

The Set In-Position Range command (INP) is used to set the NEAR signal output width (in-position check width). Up to 16 axes can be designated in an INP command block. Motion setting parameter OL□□20 (NEAR signal output width) for each designated axis is updated.
The reference range is between 1 and 65535 (reference unit).

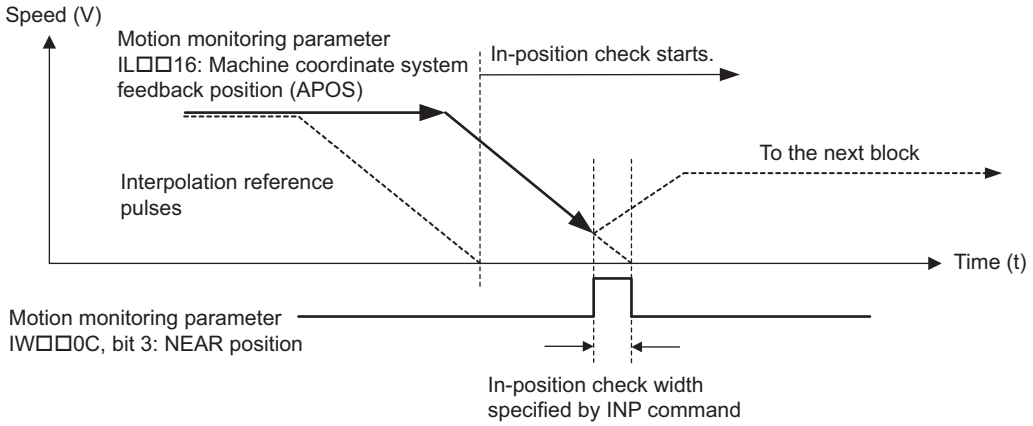


Fig. 8.50 INP Command

INFO

The SVR module does not have motion setting parameter OL□□20 (NEAR signal output width).
With the SVR module, the NEAR signal output width is treated as 0 (zero).

(2) Format

INP [*Logical axis name 1*] *NEAR signal output width* [*Logical axis name 2*] *NEAR signal output width* ... ;

Item	Unit	Usable Data
NEAR signal output width	Reference unit	· Directly designated value · Double integer type register (Indirect designation)

(3) Programming Examples

◀ EXAMPLE ▶

An INP command programming example is shown below.

```
ABS;
MOV [A1]0 [B1]0;          " Positioning to the zero point
INP [A1]100 [B1]200;      "Sets the in-position check width
MVS [A1]1000 PFN;
MVS [B1]1000 PFN;
MVS [A1]-1000 ;
END;
```

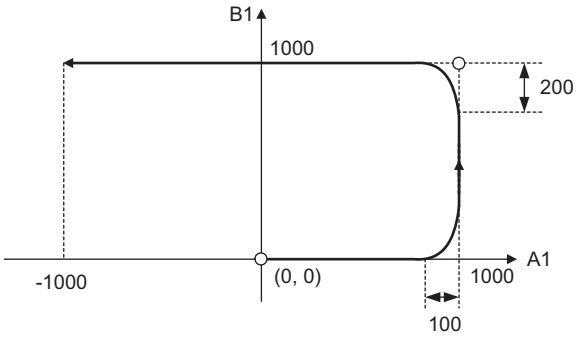


Fig. 8.51 INP Command Programming Example

8.3.6 Coordinate Plane Setting (PLN)

Motion Programs	Sequence Programs
Applicable	Not applicable

(1) Overview

The Coordinate Plane Setting command (PLN) defines two logical axes set in the parameters to designate a coordinate plane. Always execute a PLN command before executing a Circular Interpolation command (MCW, MCC) and Helical Interpolation command (MCW, MCC).
The designated coordinate plane remains in effect until it is reset by another PLN command or until the END command.

(2) Format

```
        Horizontal axis name   Vertical axis name
PLN  [Logical axis name 1]  [Logical axis name 2] ;
```

Designate two axes of a coordinate plane.

(3) Programming Examples

◀ EXAMPLE ▶

A PLN command programming example is shown below.

```
PLN[A1][B1];                          " Designates the plane composed of axes A1 and B1
MCW  [A1]50 [B1]50 R50 F1000;
```

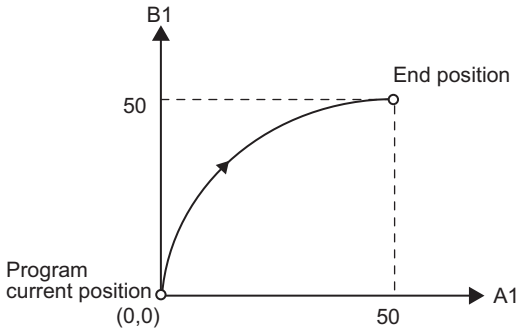
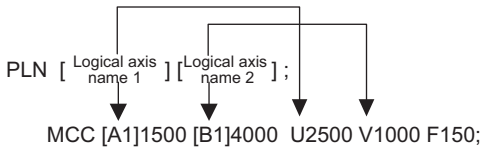


Fig. 8.52 PLN Command Programming Example

INFO

Designate an end position and a center position for circular interpolation and helical interpolation in the same order used to specify the axes in the PLN command block.



8.4 Program Control Commands

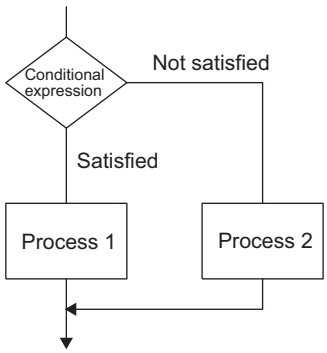
This section describes program control commands including branching commands and repeat commands.

8.4.1 Branching Commands (IF ELSE IEND)

Motion Programs	Sequence Programs
Applicable	Applicable

(1) Overview

The branching commands IF ELSE IEND execute the block between IF and ELSE when a conditional expression is satisfied. If the conditional expression is not satisfied, the block between ELSE and IEND is executed. ELSE can be omitted. If it is omitted and the conditional expression is not satisfied, execution will continue from the block after IEND.



IMPORTANT

Nesting of the branching commands IF ELSE IEND is restricted to a maximum of eight levels.

(2) Format

```
IF (conditional_expression);
 ··· (Process_1)
ELSE;
 ··· (Process_2)
IEND;
```

The conditional expressions that can be used for the branching commands are as follows.

(a) Bit Data Comparison

Format	• Use == (MATCH) command for data comparison. • Specify a register on the left, and 0 or 1 on the right. IF MB000000 == 0; "MB000000 = 0 IF MB000000 == 1; "MB000000 = 1
Operations in Conditional Expression	• &, |, and ! (AND, OR, and NOT) can be used. IF (MB000000 & MB000001) == 1; "MB000000=1 AND MB000001=1 IF (MB000000 & !MB000001) == 1; "MB000000=1 AND MB000001=0 IF (MB000000 | MB000001) == 1; "MB000000=1 OR MB000001=1 IF (MB000000 | !MB000001) == 1; "MB000000=1 OR MB000001=0
Syntax Error Examples	• When a <> (MISMATCH) is used: IF MB000000 <> 0; => Syntax error • When a numerical value is specified on the left, or a register is specified on the right: IF 1 == MB000000; => Syntax error IF MB000000 == MB000001; => Syntax error • No data comparison command: IF MB000000; => Syntax error IF (0); => Syntax error • When more than one data comparison command is used: IF (MB000000 == 0) & (MB000001 == 1); => Syntax error

(b) Integer/Double Integer/Real Number Data Comparison

Format	• All data comparison commands (==, <>, >, <, >=, <=) can be used. • Specify a register either on the left or right. IF MW00000 == 3; "MW00000 = 3 IF ML00000 <> ML00002; "ML00000 ≠ ML00002 IF 1.23456 >= MF00000; "1.23456 ≥ MF00000
Operations in Conditional Expression	• Arithmetic operations and logic operations can be used. IF MW00000 == (MW00001/3); "MW00000 = (MW00001 ÷ 3) IF (ML00000 & F0000000H) <> ML00002; "(ML00000 ∧ F0000000H) ≠ ML00002 IF 1.23456 >= (MF00000 * MF00002); "1.23456 ≥ (MF00000 × MF00002)
Syntax Error Examples	• When a constant is specified both on the left and right: IF 0 == 3; => Syntax error IF (3.14*2*1000) > 9000.0; => Syntax error • No comparison command: IF MW000000; => Syntax error IF (-1); => Syntax error • When more than one data comparison command is used: IF (MW00000 < 0) & (MW000001 > 0); => Syntax error

(3) Programming Examples

◀ EXAMPLE ▶

A programming example of branching commands (IF ELSE IEND) is shown below.

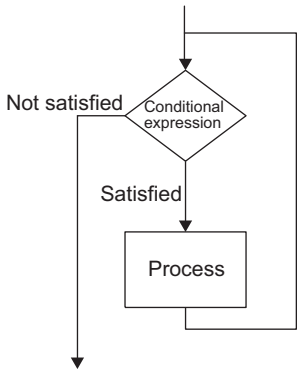
```
IF MB000000== 1;
   MOV [A1] 10000;    " If MB000000 is ON, A1 starts positioning.
ELSE;
   MOV [B1] 10000;    " If MB000000 is OFF, B1 starts positioning.
IEND;
```

8.4.2 Repeat (WHILE WEND)

Motion Programs	Sequence Programs
Applicable	Applicable

(1) Overview

The Repeat commands (WHILE WEND) repeatedly execute the blocks from WHILE and WEND as long as the conditional expression is satisfied. When the conditional expression is no longer satisfied, program execution will jump to the block after WEND.



IMPORTANT

- Nesting of the repeat commands (WHILE WEND) is restricted to a maximum of eight levels.
- If the repeated program section is created using only commands for which processing is completed in one scan, the Machine Controller may be overloaded by the scan processing, resulting in scan time over or watchdog timer error.
  When using a command for which processing is completed in one scan, be sure to enter EOX (SCAN WAIT) or TIM (DWELL TIME).
  Refer to *7.5 Commands and Execution Scans* for information on the commands for which processing is completed in one scan.

(2) Format

```
WHILE  (conditional_expression) ;
 ... ;
(Process) ;
 ... ;
WEND   ;        " End of Repeat
```

The conditional expressions that can be used for repeat commands are as follows.

(a) Bit Data Comparison

Format	• Use == (MATCH) command for data comparison. • Specify a register on the left, and 0 or 1 on the right. WHILE MB000000 == 0; "MB000000 = 0 WHILE MB000000 == 1; "MB000000 = 1
Operations in Conditional Expression	• &, |, and ! (AND, OR, and NOT) can be used. WHILE (MB000000 & MB000001) == 1; "MB000000=1 AND MB000001=1 WHILE (MB000000 & !MB000001) == 1; "MB000000=1 AND MB000001=0 WHILE (MB000000 | MB000001) == 1; "MB000000=1 OR MB000001=1 WHILE (MB000000 | !MB000001) == 1; "MB000000=1 OR MB000001=0
Syntax Error Examples	• When a <> (MISMATCH) is used: WHILE MB000000 <> 0; => Syntax error • When a numerical value is specified on the left, or a register is specified on the right: WHILE 1 == MB000000; => Syntax error WHILE MB000000 == MB000001; => Syntax error • No data comparison command: WHILE MB000000; => Syntax error WHILE (0); => Syntax error • When more than one data comparison command is used: WHILE (MB000000 == 0) & (MB000001 == 1); => Syntax error

(b) Integer/Double Integer/Real Number Data Comparison

Format	• All data comparison commands (==, <>, >, <, >=, <=) can be used. • Specify a register either on the left or right. WHILE MW00000 == 3; "MW00000 = 3 WHILE ML00000 <> ML00002; "ML00000 ≠ ML00002 WHILE 1.23456 >= MF00000; "1.23456 ≥ MF00000
Operations in Conditional Expression	• Arithmetic operations and logic operations can be used. WHILE MW00000 == (MW00001/3); "MW00000 = (MW00001 ÷ 3) WHILE (ML00000 & F0000000H) <> ML00002; "(ML00000 ∧ F0000000H) ≠ ML00002 WHILE 1.23456 >= (MF00000 * MF00002); "1.23456 ≥ (MF00000 × MF00002)
Syntax Error Examples	• When a constant is specified both on the left and right: WHILE 0 == 3; => Syntax error WHILE (3.14*2*1000) > 9000.0; => Syntax error • No data comparison command WHILE MW000000; => Syntax error WHILE (-1); => Syntax error • When more than one data comparison command is used: WHILE (MW00000 < 0) & (MW000001 > 0); => Syntax error

(3) Programming Examples

A programming example of a repeat command (WHILE WEND) is shown below.
With this program example, 10 circles of radius 50 will be drawn.

◀ EXAMPLE ▶

```
MOV [A1] 0 [B1] 0;                    "Positioning
MW00100 = 1;                          "Counter preset
INC;                                  "Incremental mode designation
PLN [A1] [B1];                        "Coordinate plane designation
WHILE MW00100 <= 10 ;                 "Repeat command
   MCW [A1]0 [B1]0 U50. V50. F8000 ;  "Circular interpolation
   MOV [A1]50. [B1]50.;               "Positioning
   MW00100 = MW00100 + 1;             "Counter increments
WEND ;                                "End of repeat programming
```

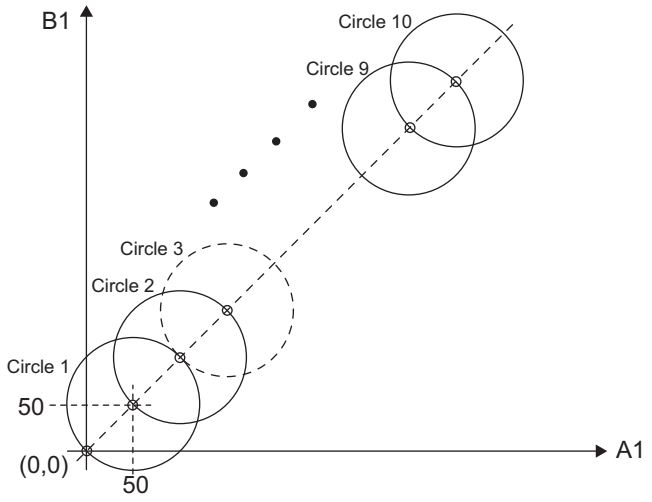


Fig. 8.53 Repeat Commands (WHILE WEND) Programming Example

8.4.3 Parallel Execution (PFORK, JOINTO, PJOINT)

Motion Programs	Sequence Programs
Applicable	Not applicable

(1) Overview

The Parallel Fork (PFORK) command performs parallel execution for blocks with the designated labels. After each parallel process has been executed, execution is merged at the label designated by the JOINTO command. A maximum of four parallel processes can be designated.

For further in formation on the labels, refer to *7.1.2 (1) Label.*

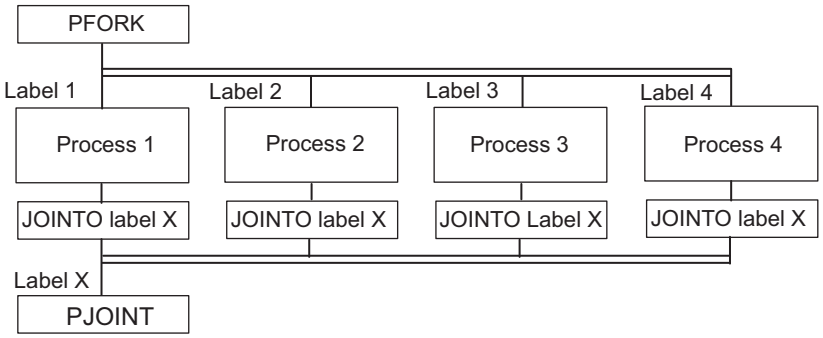


Fig. 8.54 Designating Parallel Execution Commands (PFORK, JOINTO, PJOINT)

With the above commands, the labelled blocks (process1, process2, process3,...) designated by the PFORK command are executed in parallel. After each parallel process has been executed, execution is merged at the label designated by the JOINTO commands.

These commands enable the designation of any combination of commands for parallel execution, such as axis move commands and sequence commands, or axis move commands and another axis move commands.

(a) Commands Designated Before PFORK

Values set by commands designated before the PFORK command, e.g., FMX, ABS/INC, F designation, IFP, PLN, IAC/IDC, are effective in processes executed in parallel for the parallel execution commands. Commands can also be used to set different values in each of the parallel processes. After merging, processing will continue using the values set in the leftmost process.

(b) Parallel Execution Commands in Subprograms

The following restrictions apply to the parallel execution commands in subprograms:

- A maximum of two parallel processes are possible in a subprogram.
- An MSEE command can be coded only in the blocks designated by the first label.

```
      PFORK 0001 0002;
0001:MVS [A1]100.[C1]100.;
       JOINTO 0003;
0002:IOW MW10000==1;
      JOINTO 0003;
0003:PJOINT;
```

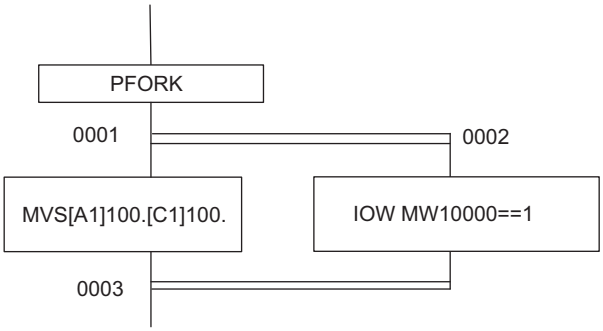


Fig. 8.55 Parallel Execution Commands in Subprograms

IMPORTANT

- If the same label is used more than once in a program, an error will result (“Duplicate labels are defined”).
- If the number of PFORK branches and the number of labels are different, an error will result.

(2) Format

```
PFORK   Label1    Label 2    Label 3 .....

    Label 1: Process 1
                JOINTO   Label X;
    Label 2: Process 2
                JOINTO   Label X;
    Label 3: Process 3
                JOINTO   Label X;
    Label X: PJOINT
```

(3) Programming Examples

◀ EXAMPLE ▶

A programming example of parallel execution commands (PFORK, JOINTO, PJOINT) is shown below.

```
    MOV [A1]100. [B1]150.;
    MVS [A1]200. [B1]250. F1000;
    PFORK 0001 0002 0003;
0001:MVS [A1]300. [B1]100.
    JOINTO 0004;
0002:MW12345=MW10000+MW10002;
    IOW MB120001==1;
    JOINTO 0004;
0003:MVS [C1]100. [D1]100. F3000;
    JOINTO 0004;
0004:PJOINT;
    MOV [A1]500. [B1]500. [C1]500.;
        •
        •
```

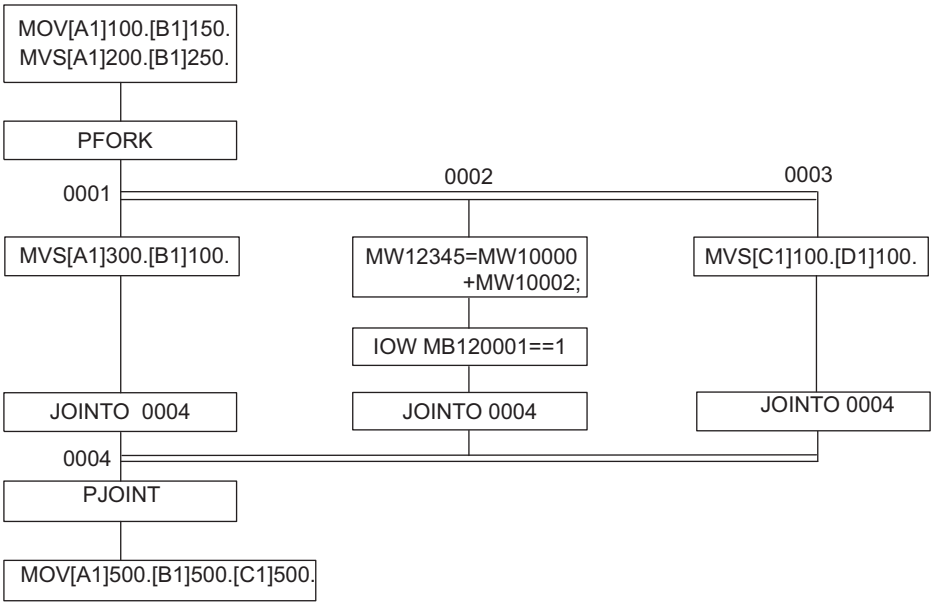


Fig. 8.56 Parallel Execution Command (PFORK, JOINTO, PJOINT) Programming Example

8.4.4 Selective Execution (SFORK, JOINTO, SJOINT)

Motion Programs	Sequence Programs
Applicable	Applicable

(1) Overview

The Selective Execution commands (SFORK, JOINTO, SJOINT) execute labelled blocks following question marks (?) when the designated conditional expressions are satisfied. After each parallel process has been executed, execution is merged at the block with the label designated by the JOINTO commands. Up to 16 conditional expressions including DEFAULT can be designated.
If not all the designated conditional expressions are satisfied, the labelled block following DEFAULT? is executed.
DEFAULT can be designated only in the last conditional expression.
DEFAULT designation can be omitted in motion programs, but not in sequence programs.

INFO

- With MP2000 series Machine Controller, the system version number must be Ver.2.66 or later to use the SFORK command in sequence programs. There is no limitation in the system version number for using the SFORK command in motion programs.
- The following versions of programming tool MPE720 are required to designate DEFAULT.

MP2000 Series Machine Controller	Required Version
All models	Any version number

MPE720	Required Version
MPE720 Ver.5	MPE720 Ver.5.41 or later
MPE720 Ver.6	MPE720 Ver.6.06 or later MPE720 Ver.6.06 Lite or later

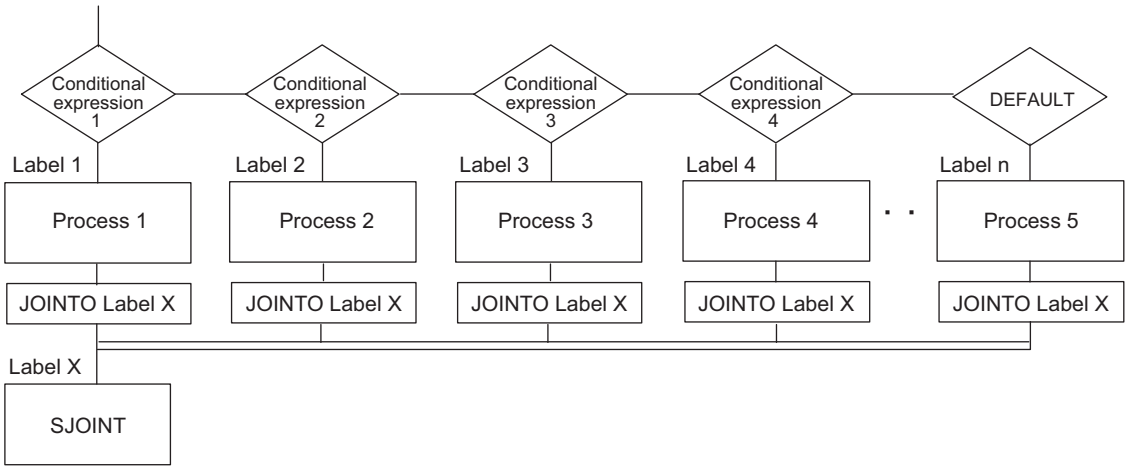


Fig. 8.57 Designating Selective Execution Commands (SFORK, JOINTO, SJOINT)

INFO

- The conditional expressions are examined in order from conditional expression 1. Even when more than one conditional expression is satisfied, processing is executed from the label that first satisfies the conditional expression.
- When using an SFORK command in motion programs, be sure to code conditions that will be satisfied. If a condition is not satisfied, processing will remain in wait status at the SFORK command block until the condition is satisfied.

(2) Format

```
SFORK  Conditional expression 1 ? Label1, Conditional expression 2 ? Label 2, Conditional expression 3 ?
Label 3, Conditional expression 4 ? Label 4,
....., DEFAULT? Label n ;

    Label 1: Process 1
                JOINTO   Label X
    Label 2: Process 2
                JOINTO   Label X
    Label 3: Process 3
                JOINTO   Label X
    Label 4: Process 4
                JOINTO   Label X
                •
                •
    Label n: Process n
                JOINTO   Label X
    Label X:  SJOINT
```

The conditional expressions that can be used for selective execution command (SFORK) are as follows.

(a) Bit Type Data Comparison

Format	• Use == (MATCH) command for data comparison. • Specify a register on the left, and 0 or 1 on the right. MB000000 == 0? Label "MB000000 = 0 MB000000 == 1? Label "MB000000 = 1
Operations in Conditional Expressions	• &, |, and ! (AND, OR, and NOT) can be used. (MB000000 & MB000001) == 1? Label "MB000000 =1 AND MB000001 = 1 (MB000000 & !MB000001) == 1? Label "MB000000 =1 AND MB000001 = 0 (MB000000 | MB000001) == 1? Label "MB000000 =1 OR MB000001 = 1 (MB000000 | !MB000001) == 1? Label "MB000000 =1 OR MB000001 = 0
Syntax Error Examples	• When <> (MISMATCH) is used: MB000000 <> 0? Label => Syntax error • When a numerical value is specified on the left, or a register is specified on the right: 1 == MB000000? Label => Syntax error MB000000 == MB000001? Label => Syntax error • No data comparison command: MB000000? Label => Syntax error (0)? Label => Syntax error • When more than one data comparison command is used: (MB000000 == 0) & (MB000001 == 1)? Label => Syntax error

(b) Integer/Double Integer/Real Number Data Comparison

Format	• All data comparison commands (==, <>, >, <, >=, <=) can be used. • Specify a register either on the left or right. MW00000 == 3? Label "MW00000 = 3 ML00000 <> ML00002? Label "ML00000 ≠ ML00002 1.23456 >= MF00000? Label "1.23456 ≥ MF00000
Operations in Conditional Expression	• Arithmetic operation and logic operation can be used. MW00000 == (MW00001/3)? Label "MW00000 = (MW00001 ÷ 3) (ML00000 & F0000000H) <> ML00002? Label "(ML00000 ∧ F0000000H) ≠ ML00002 1.23456 >= (MF00000 * MF00002)? Label "1.23456 ≥ (MF00000 MF00002)
Syntax Error Examples	• When a constant is specified both on the left and right: 0 == 3? Label => Syntax error (3.14*2*1000) > 9000.0? Label => Syntax error • No data comparison command: MW000000? *Label* => Syntax error (-1)? *Label* => Syntax error • When more than one data comparison command is specified (MW00000 < 0) & (MW000001 > 0)? Label => Syntax error

(3) Programming Examples

A programming example of selective execution commands (SFORK, JOINTO, SJOINT) is shown below.

◀ EXAMPLE ▶

```
    MOV [A1]100.[B1]150.;
    MVS [A1]200.[B1]250.F1000;
    SFORK  MW00100==1 ? 0001,MW00100==2 ? 0002,MW00100==3 ? 0003,DEFAULT ? 0004;
0001:MVS [A1]300.[B1]100.F3000;
    JOINTO 0005
0002:MVS [A1]300.[C1]100.F3000;
    JOINTO 0005
0003:MVS [C1]300.[S]100.F3000;
    JOINTO 0005
0004:JOINTO 0005;
0005:SJOINT;
MOV[A1]500.[B1]500.[C1]500.
```

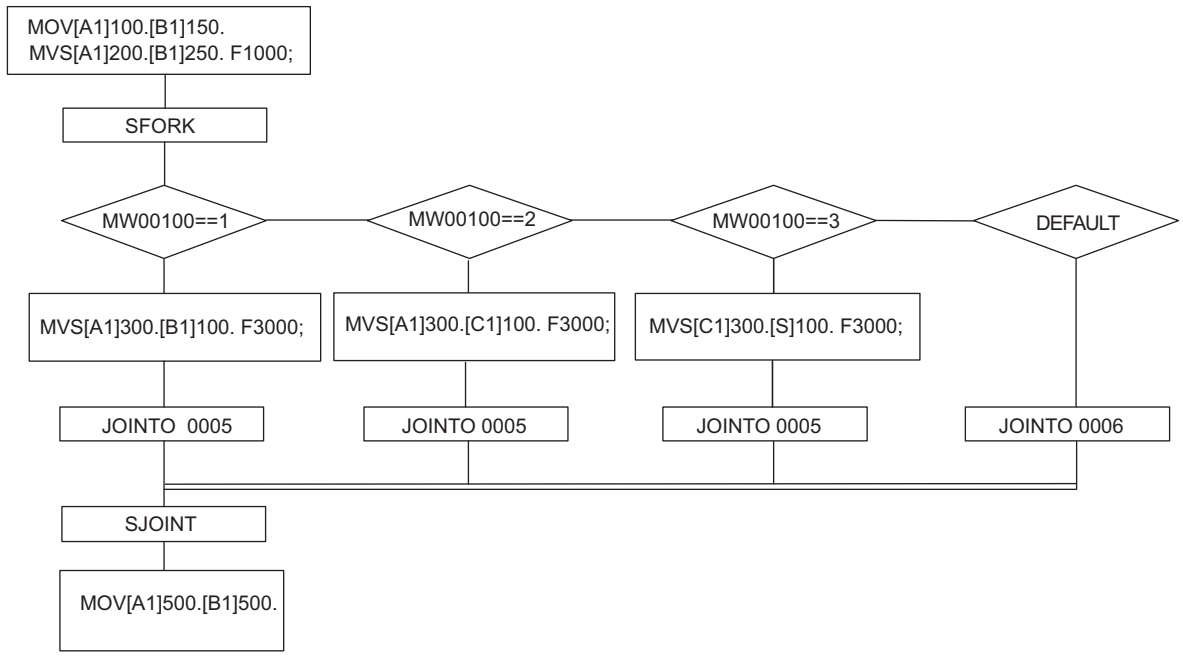


Fig. 8.58 Selective Execution Command (SFORK, JOINT, SJOINT) Programming Example

8.4.5 Motion Subprogram Call (MSEE)

Motion Programs	Sequence Programs
Applicable	Not applicable

(1) Overview

The Motion Subprogram Call command (MSEE) can call a subprogram that has been stored in the motion program memory from the motion program.
Nesting of the subprogram call command (MSEE) is restricted to a maximum of eight levels.

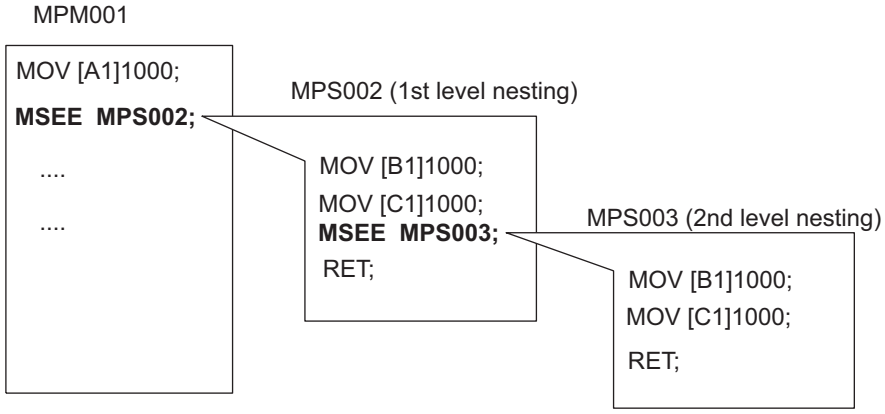


Fig. 8.59 Subprograms

A subprogram return command (RET) must be coded at the end of the subprogram.

IMPORTANT

■ Restrictions on Subprograms

The following restrictions apply to motion program coding within a subprogram.

- A maximum of two parallel executions with a PFORK command
- If the program number of a main program is called by an MSEE command, the program will not be executed.

(2) Format

MSEE MPS Subprogram number;

Item	Unit	Usable Data
Subprogram number	–	A number between 001 and 256

(3) Programming Examples

A programming example of the MSEE command to call the motion subprogram MPS101 is shown below.

◀ EXAMPLE ▶

```
MSEE  MPS101;
      Subprogram number designation
```

8.4.6 Sequence Subprogram Call (SSEE)

Motion Programs	Sequence Programs
Not applicable	Applicable

(1) Overview

The Sequence Subprogram Call command (SSEE) calls a subprogram that has been stored in the sequence program memory from the sequence program.
Nesting of the subprogram call command is restricted to a maximum of eight levels.

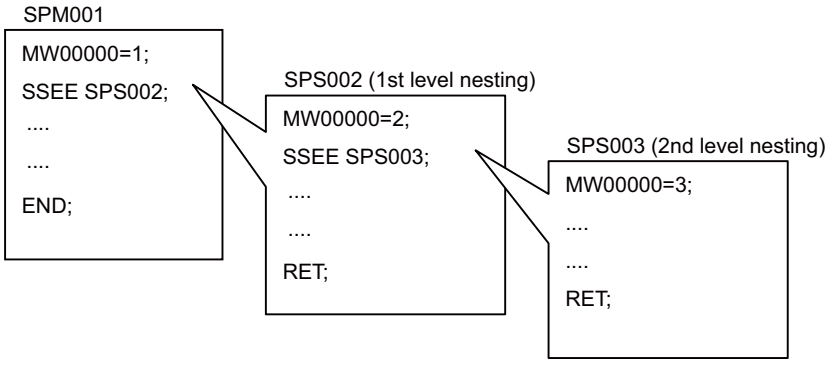


Fig. 8.60 Subprograms

A subprogram return command (RET) must be coded at the end of a subprogram.

IMPORTANT

■ Restrictions on Subprograms

The following restriction applies to sequence program coding within a subprogram.

- If a main program is called by the SSEE command, the program will not be executed.

(2) Format

SSEE SPS Subprogram number ;

Item	Unit	Usable Data
Subprogram number	–	A number between 001 and 256

(3) Programming Examples

A programming example of the SSEE command to call the subprogram SPS101 is shown below.

◀ EXAMPLE ▶

```
SSEE  SPS101;
      Subprogram number designation
```

8.4.7 User Function Call From Motion Program (UFC)

Motion Programs	Sequence Programs
Applicable	Not applicable

(1) Overview

The User Function Call From Motion Program command (UFC) calls a user function (ladder program) from a motion program.
When execution of the called user function is completed, the block after the UFC command block will be executed.

IMPORTANT

For the user function called from the motion program, output bit YB000000 is used for completion judgement (Complete Bit).

- When YB000000 = OFF at the completion of execution of user function:
- Execution of the user function is recognized as incomplete, and the user function will be called again during the next scan.
- When YB000000 = ON at the completion of execution of user function:
- Execution of the user function is recognized as complete, and the block after the UFC command block will be executed.

(2) Format

UFC *Function name* *Input data*, *Input address*, *Output data*;

Item	Unit	Usable Data
Function name	–	ASCII 8 bytes
Input data	–	Max: 16 data items (Minimum: 1 data item)
Input address	–	Max: 1 address
Output data *	–	Max:16 data items (Minimum: 1 data item)

* The input address can be omitted. [UFC Input data, Output data;] is used if there is no input address. The minimum requirement is one input data item and one output data item.

(3) Programming Examples

A UFC command programming example is shown below.

◀ EXAMPLE ▶

```
UFC KANSUU  MB000000 IW0010 MB000002, MA00100, MB000001 MW00200 ML00201;
     Function      Input data          Input address        Output data
      name
```

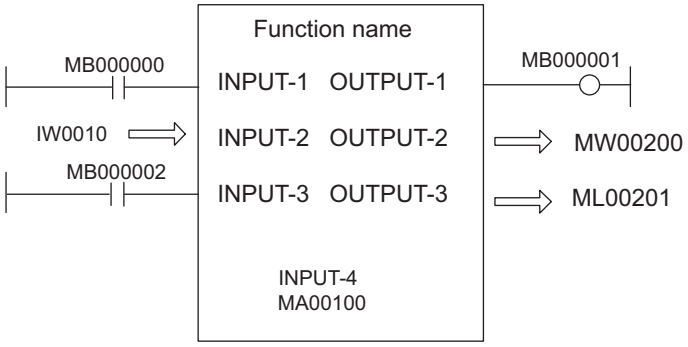


Fig. 8.61 User Function Call Command (UFC)

(4) UFC Command Creation Procedure

Use the following procedure to create a UFC command.

Determine the UFC command specifications.	• Determine the number of I/Os and data type. • Determine the function name.
⇩	
Set the following on the Program Property window. • Configuration definition • I/O definitions	Use the MPE720 to enter the definitions.
⇩	
Create the user functions (ladder program).	Create the user functions in the same way as for the drawings, except for the register type to be used.
⇩	
Create the motion program.	Write a UFC command block with the format: UFC function name input data, input address, output data;
⇩	
Check the operation.	

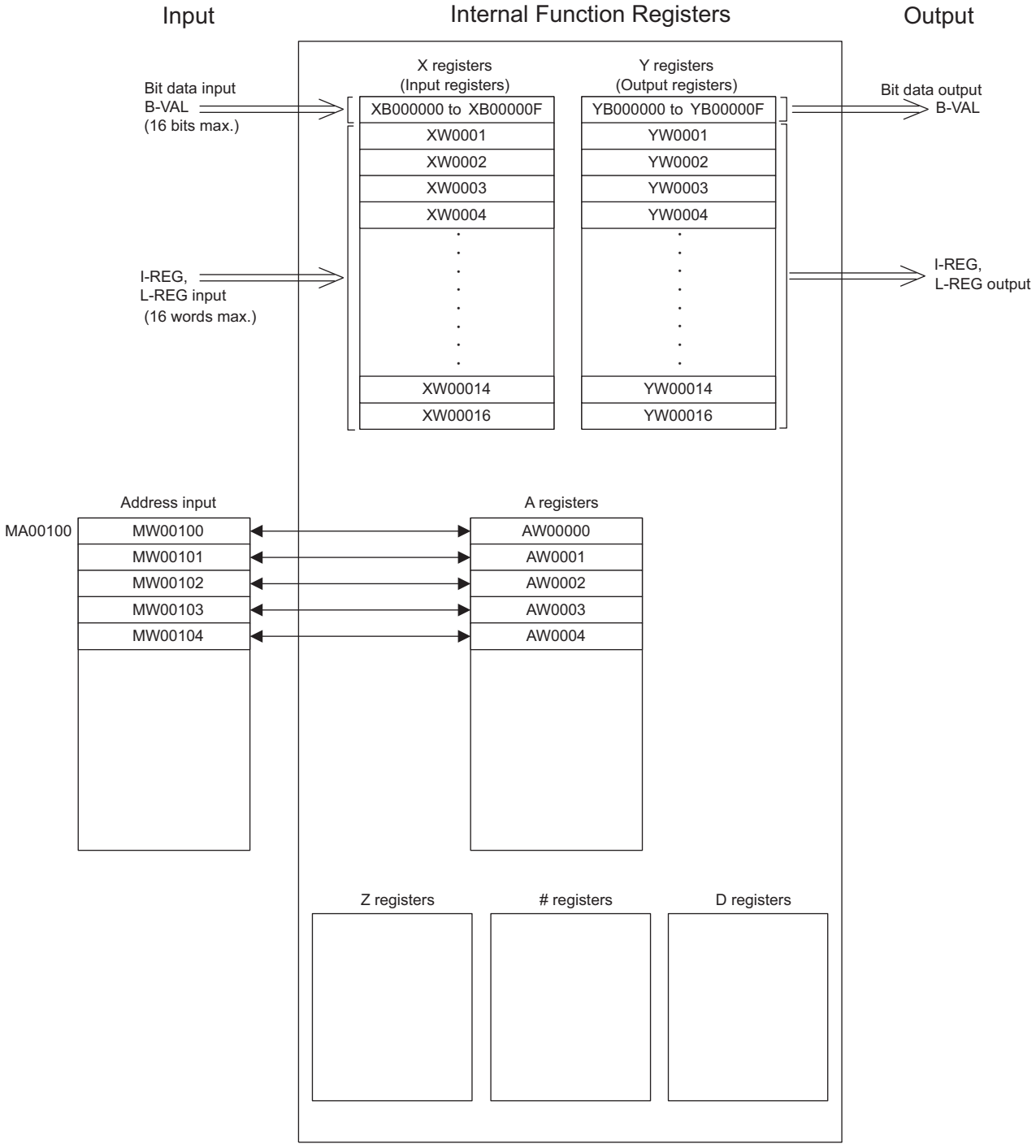
(5) Register Types Used Within User Functions

The data types are as follows.

Data Type	
BIT	Bit
WORD	Integer
LONG	Double integer
FLOAT	Real number

(6) Relationship between I/O Registers and Internal Function Registers

The correspondence between the I/O registers designated by the UFC command and the function registers is shown below.



The following 11 types of register can be used in each function.

Table 8.2 Function Registers

Type	Name	Designation Method	Description	Characteristics
X	Function input register	XB, XW, XL, XFnnnnn	Input to a function Bit input: XB000000 to XB0000F Integer input: YW00001 to XW00016 Double integer input: XL00001 to XL00015 Register number nnnnn is a decimal expression.	Unique to each function
Y	Function output register	YB, YW, YL, YFnnnnn	Input to a function Bit input: YB000000 to YB0000F Integer input: YW00001 to YW00016 Double integer input: YL00001 to YL00015 Register number nnnnn is a decimal expression	
Z	Internal function register	ZB, ZW, ZL, ZFnnnnn	Internal register unique to each function Can be used by the function for internal processes. Register number nnnnn is a decimal expression.	
A	External function register	AB, AW, AL, AFnnnnn	External registers that use the address input value as the base address. For linking with S, M, I, O, #, and DAnnnnn Register number nnnnn is a decimal expression.	
#	# register	#B, #W, #L, #Fnnnnn (#Annnnn)	Registers that can be referenced only by a program. Can be referenced only the corresponding drawing. The actual range to be used is specified by the user using the MPE720. Register number nnnnn is a decimal expression.	
D	D register	DB, DW, DL, DFnnnnn (DAnnnnn)	Registers unique to each drawing Can be referenced only by the corresponding drawing. The actual range to be used is specified by the user using the MPE720. Register number nnnnn is a decimal expression.	
S	System register	SB, SW, SL, SFnnnnn (SAnnnnn)	Same as the drawing registers. These registers are used for both drawings and functions. Care must be taken in using them to reference the same function from drawings with different priority levels.	Common to all drawings
M	Data register	MB, MW, ML, MFnnnnn (MAnnnnn)		
I	Input register	IB, IW, IL, IFhhhh (IAhhhh)		
O	Output register	OB, OW, OL, OFhhhh (OAhhhh)		
C	Constant register	CB, CW, CL, CFnnnnn (CAnnnnn)		

Note: SA, MA, IA, OA, DA, #A, and CA can also be used inside functions.

An example of the transfer of I/O registers is shown below.

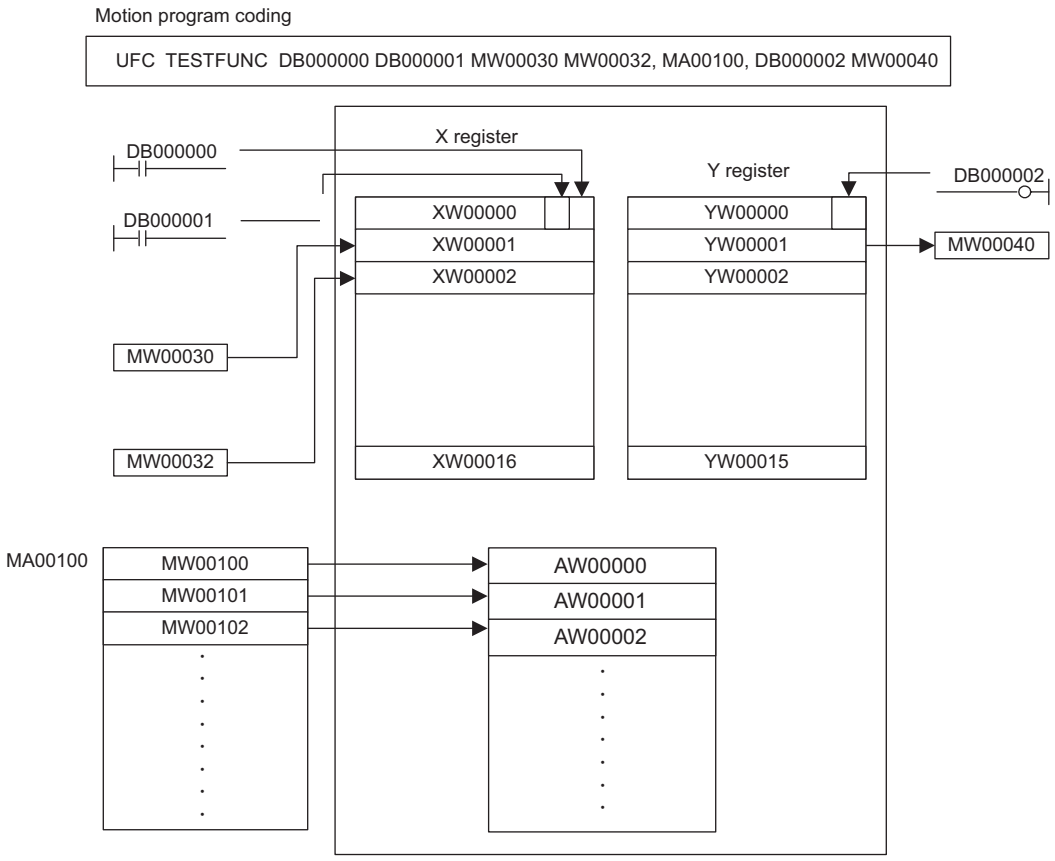


Fig. 8.62 Motion Program Coding

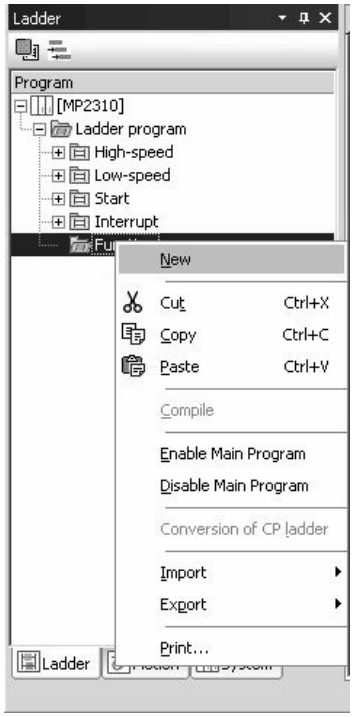
(7) Creating User Functions

The procedure for creating the user function of the following specifications is described below.

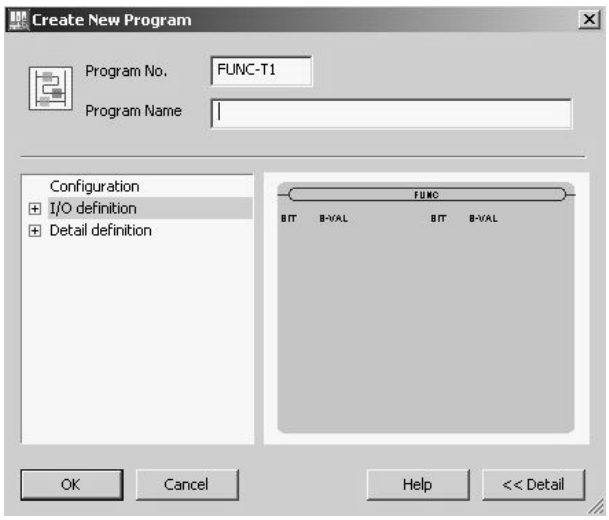
Specifications	Motion Program
Designate the servo axis No. and speed data, and set this in the motion setting parameter OL□□10: Speed reference setting.	MW00030 = Servo axis No. (1 or 2) ML00032 = Rapid traverse speed UFC FUNC-T1 MW00030 ML00032,,DB000001;

Use the following procedure to create the user function.

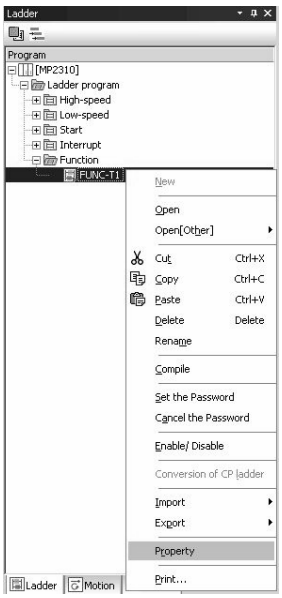
1. Open the **Ladder** subwindow. Right-click **Function** under **Ladder program**. Click **New** in the pop-up menu that will appear.



2. Enter *FUNC-T1* in the **Program Name** input field in the **Create New Program** dialog box, and then click **OK**.

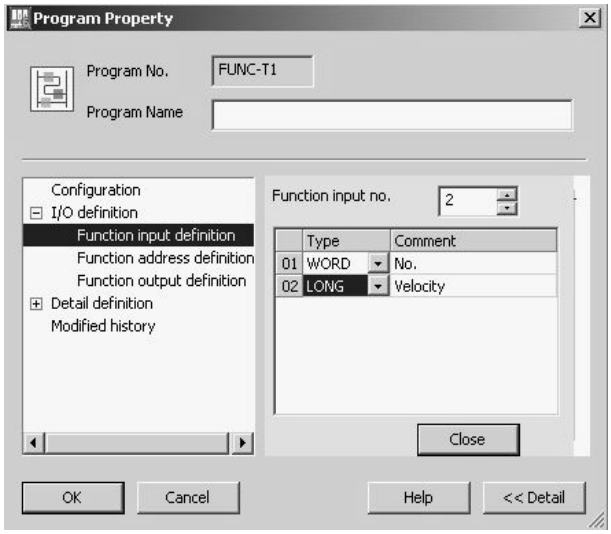


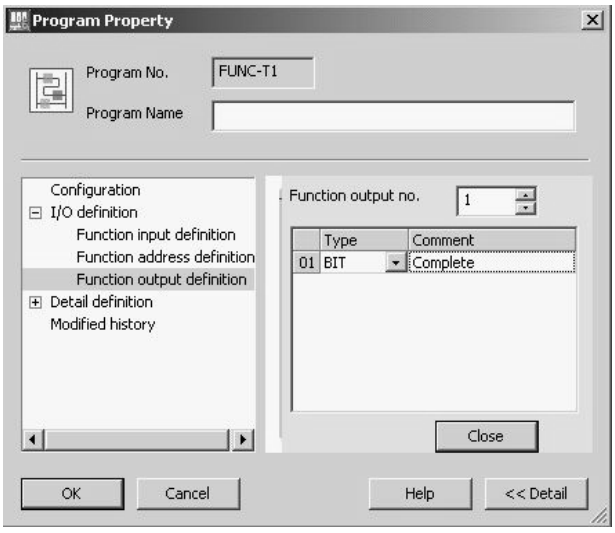
3. A blank ladder program field will appear in the **Ladder** subwindow. Right-click **FUNC-T1** to select **Property** from the pop-up menu.



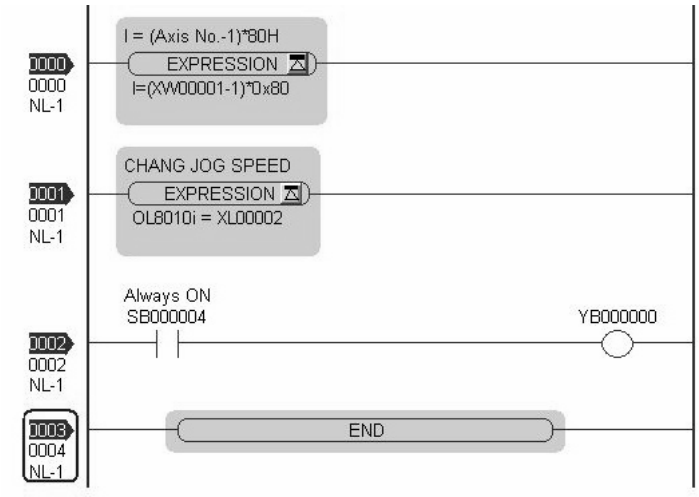
4. In the **Program Property** window, click **Function input definition** under **I/O definition** to set the number of function inputs and data type, and click **Function output definition** to set the number of function outputs and data type.

Example: For the UFC command block UFC FUNC-T1 MW00030 ML00032,,DB000001;,
the settings in the **Program Property** window are as shown below.

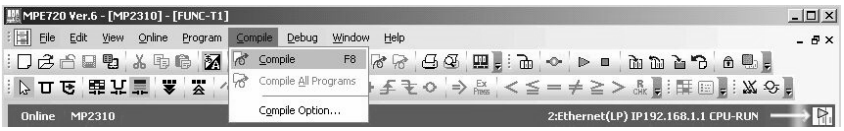




5. Close the **DWG Configuration Definition** window, and edit the user function program in the ladder program edit window.



6. From the **Compile** menu, select **Compile F8**.



7. In the **Motion Editor** window, create a program to call user function FUNC-T1.

```
MW00030 = 1;
ML00032 = 500;
UFC FUNC-T1 MW00030 ML00032, , DB000001;
END;
```

The user function that is called from the motion program is created.
Execute the motion program to check the operation.

8.4.8 User Function Call from Sequence Program (FUNC)

Motion Programs	Sequence Programs
Not applicable	Applicable

(1) Overview

The User Function Call from Sequence Program command (FUNC) calls user functions (ladder programs) from the sequence program.

(2) Format

UFC Function name Input data 1 Input data 2 Input data 3 ....., Input address,

Output data 1 Output data 2 Output data 3 ..... ;

Item	Unit	Usable Data
Function name	–	ASCII 8 bytes
Input data	–	Max: 16 data items (minimum: 1 data item)
Input address	–	Max: 1 address
Output data	–	Max: 16 data items (minimum: 1 data item)

Note: 1. More than one item for input data and output data can be described.
(At least one item each for input data and output data must be described.) The input address can be omitted.
When the input address is omitted, describe only the comma (,).
2. The FUNC command calls a user function. The execution proceeds to the next block after the FUNC command whether or not the user function execution has been completed.

(3) Programming Examples

A FUNC command programming example is shown below.

In this example, three input data items, one input address, and three output data items are described.

◀ EXAMPLE ▶

FUNC KANSUU MB000000 IW0010 MB000020, MA00100, MB000001 MW00201 ML00202;
Function name — Input data — Input address — Output data

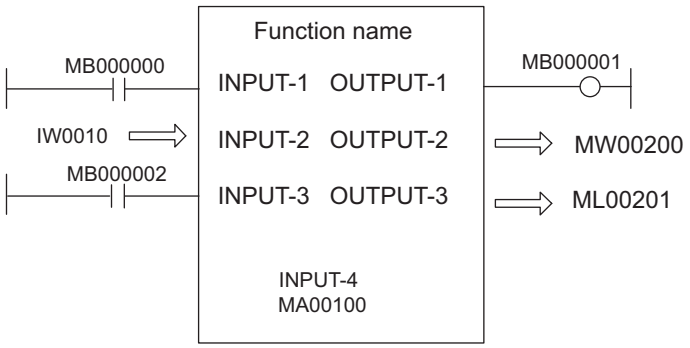


Fig. 8.63 User Function Call Command (FUNC) Programming Example

8.4.9 Program End (END)

Motion Programs	Sequence Programs
Applicable	Applicable

(1) Overview

The Program End command (END) ends program operation.
No other commands can be coded in the same block as the END command.
Program operation ends after execution of the END command block.
If there is a move command in the previous block, the program operation ends after the in-position check is completed.

(2) Format

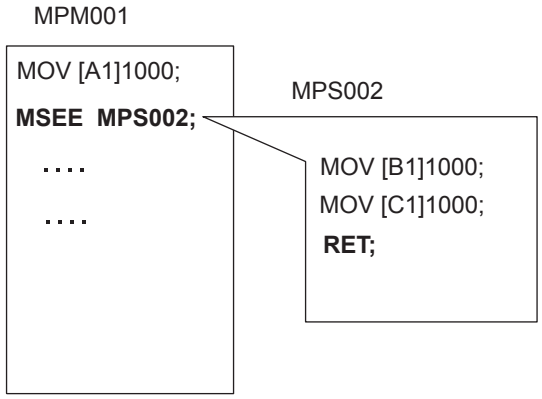
END;
Program end

8.4.10 Subprogram End (RET)

Motion Programs	Sequence Programs
Applicable	Applicable

(1) Overview

The Subprogram End command (RET) ends subprogram operation.
After operation of the called subprogram is ended by the RET command, execution proceeds to the block after the Motion Subprogram Call command (MSEE) or Sequence Subprogram Call command (SSEE) in the main program or subprogram that called the subprogram.



(2) Format

RET;
Subprogram end

8.4.11 Dwell Time (TIM)

Motion Programs	Sequence Programs
Applicable	Not applicable

(1) Overview

The Dwell Time command (TIM) causes execution to pause for a specified period of time before the start of execution of the next command block.
A period of time between 0.00 to 600.00 seconds can be specified.

(2) Format

TIM T Dwell time ;

Item	Unit	Usable Data
Dwell time	0.01 s	· Directly designated value · Integer type register (Indirect designation)

(3) Programming Examples

A TIM command programming example is shown below.

◀ EXAMPLE ▶

```
MOV [A1]100;
TIM T250 ;
      2.5 seconds
```

The TIM command is executed after positioning has been completed.

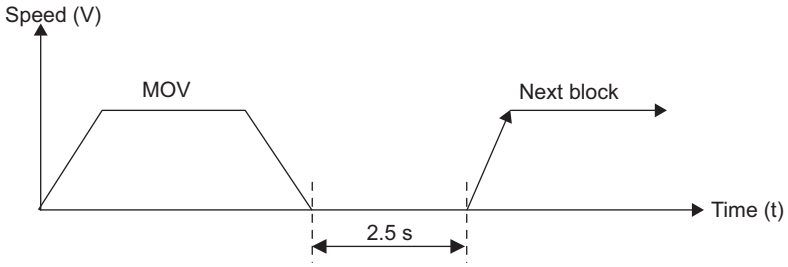


Fig. 8.64 Dwell Time Command (TIM) Programming Example

8.4.12 I/O Variable Wait (IOW)

Motion Programs	Sequence Programs
Applicable	Not applicable

(1) Overview

The I/O Variable Wait command (IOW) causes execution to wait until the status specified by the conditional expression is satisfied. When the condition is satisfied, execution proceeds to the next block.

(2) Format

IOW IB00001&IB00002 == 1;

A

Item	Description	Usable Data
A	Conditional expression	• All integer type, double integer type, and real number type registers (Excluding # and C registers) • The above registers with subscript • Subscript registers • Constants

The conditional expressions that can be used in the IOW command block are as follows.

(a) Bit Data Comparison

Formant	• Use = = (MATCH) command for data comparison. • Specify a register on the left, and 0 or 1 on the right. IOW MB000000 == 0; "MB000000 = 0 IOW MB000000 == 1; "MB000000 = 1
Operations in Conditional Expression	• &, |, and ! (AND, OR, NOT) can be used. IOW (MB000000 & MB000001) == 1; "MB000000 =1 AND MB000001 = 1 IOW (MB000000 & !MB000001) == 1; "MB000000 =1 AND MB000001 = 0 IOW (MB000000 | MB000001) == 1; "MB000000 = 1 OR MB000001 = 1 IOW (MB000000 | !MB000001) == 1; "MB000000 = 1 OR MB000001 = 0
Syntax Error Examples	• When <> (MISMATCH) is used: IOW MB000000 <> 0; => Syntax error • When a numerical value is specified on the left, or a register is specified on the right: IOW 1 == MB000000; => Syntax error IOW MB000000 == MB000001; => Syntax error • No data comparison command: IOW MB000000; => Syntax error IOW (0); => Syntax error • When more than one data comparison command is used: IOW (MB000000 == 0) & (MB000001 == 1); => Syntax error

(b) Integer/Double Integer/Real Number Data Comparison

Format	• All data comparison commands (==, <>, >, <, >=, <=) can be used. • Specify a register either on the left or right. IOW MW00000 == 3; "MW00000 = 3 IOW ML00000 <> ML00002; "ML00000 ≠ ML00002 IOW 1.23456 >= MF00000; "1.23456 ≥ MF00000
Operations in Conditional Expression	• Arithmetic operations and logic operations can be used. IOW MW00000 == (MW00001/3); "MW00000 = (MW00001 ÷ 3) IOW (ML00000 & F0000000H) <> ML00002; "(ML00000 ∧ F0000000H) ≠ ML00002 IOW 1.23456 >= (MF00000 * MF00002); "1.23456 ≥ (MF00000 × MF00002)
Syntax Error Examples	• When a constant is specified both on the left and right: IOW 0 == 3; => Syntax error IOW (3.14*2*1000) > 9000.0; => Syntax error • No data comparison command: IOW MW000000; => Syntax error IOW (-1); => Syntax error • When more than one data comparison command is used: IOW (MW00000 < 0) & (MW000001 > 0); => Syntax error

(3) Programming Examples

An IOW command programming example is shown below.

◀ EXAMPLE ▶

```
IOW (MB001001&MB001002)== 1;
MOV [A1]1000;
```

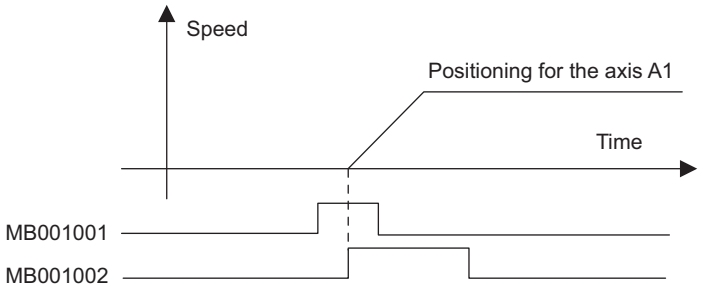


Fig. 8.65 I/O Variable Wait Command (IOW) Programming Example

8.4.13 One Scan Wait (EOX)

Motion Programs	Sequence Programs
Applicable	Not applicable

(1) Overview

The One Scan Wait command (EOX) causes program execution to be suspended for one scan.
The block after EOX command will be executed in the next scan.

(2) Format

```
EOX;
```

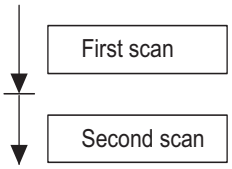
(3) Programming Examples

An EOX command programming example is shown below.

(a) Used in Combination with Sequence Commands

◀ EXAMPLE ▶

```
MW00000=100;
OB00010=1;
EOX;
OB00011=0;
```



(b) Used with a WHILE Command

◀ EXAMPLE ▶

```
WHILE  OB00010==1;
EOX;
WEND;
```

8.4.14 Single-block Signal Disabled (SNGD)/Single-block Signal Enabled (SNGE)

Motion Programs	Sequence Programs
Applicable	Not applicable

(1) Overview

The Single-block Signal Disabled command (SNGD) and the Single-block Signal Enabled command (SNGE) are used to specify whether to disable or enable the single step operation when debugging a program.
Regardless of the single-block operation mode setting, the blocks enclosed between SNGD and SGNE commands are executed continuously without single-block stops.

TERMS

■ Single-block operation mode

In single-block operation mode, a stop is executed for each block.

(2) Format

```
SNGD;
Blocks to be continuously executed
SNGE;
```

(3) Programming Examples

◀ EXAMPLE ▶

The programming example of SNGD and SNGE commands is shown below.

```
MVS  [A1]0  [B1]0;
SNGD;
MVS  [A1]100  [B1]200;   " 1 "
MB000101 = 1;            " 2 "
MB000102 = 1;            " 3 "
SNGE;
MB000103 = 1;
```

In the above example, the blocks 1 to 3 enclosed between SNGD and SNGE commands are executed continuously without single-block stops regardless of the single-block operation mode setting.

8.5 Arithmetic Operations

This section explains the arithmetic operation commands.
For priority levels of Arithmetic Operation, refer to *7.4 Priority Levels of Operations*.

8.5.1 Substitute (=)

Motion Programs	Sequence Programs
Applicable	Applicable

(1) Overview

The operation result on the right side of the expression is substituted in the register on the left side.

(2) Format

Result = *Math expression* ;

A B

Item	Description	Usable Data
A	Result	• All bit type, integer type, double integer type, real number type registers (Excluding # and C registers) • The above registers with subscript • Subscript registers
B	Math expression	• All bit type, integer type, double integer type, real number type registers (Excluding # and C registers) • The above registers with subscript • Subscript registers • Constants

(3) Programming Examples

Substitute (=) command programming examples are shown below.

◀ EXAMPLE ▶

Type	Motion Programs/ Sequence Programs	Ladder Programs
B	MB001000=1;	
W	MW00100=12345;	
L	ML00100=1234567;	
F	MF00100=1.2345;	

8.5.2 Add (+)

Motion Programs	Sequence Programs
Applicable	Applicable

(1) Overview

ADD (+) performs integer and real number addition on the right side and stores the result in the register on the left side. With mixed integers and real numbers, the data type on the left side is also stored.

(2) Format

MW00101 = MW00100 + 12345 ;

A B C

Item	Description	Usable Registers
A	Data output	• All integer type, double integer type, real number type registers (Excluding # and C registers) • The above registers with subscript • Subscript registers
B	Data input	• All integer type, double integer type, real number type registers (Excluding # and C registers) • The above registers with subscript • Subscript registers • Constants
C	Data to add	

(3) Programming Examples

Add (+) command programming examples are shown below.

◀ EXAMPLE ▶

Type	Motion Programs/ Sequence Programs	Ladder Programs
B	–	–
W	MW00101=MW00100+12345;	
L	ML00106=ML00102+ML00104;	
F	MF00202=MF00200+1.23456;	

IMPORTANT

With an operation where the variables are of different data types, the result will be stored according to the data type on the left side.

For details, refer to *6.1.2 Global Variables and Local Variables*.

8.5.3 Subtract (-)

Motion Programs	Sequence Programs
Applicable	Applicable

(1) Overview

Subtract (-) performs integer and real number subtraction on the right side and stores the result in the register on the left side. With mixed integers and real numbers, the data type on the left side is stored.

(2) Format

MW00101 = MW00100 -12345 ;

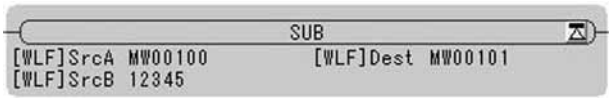
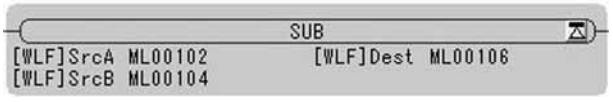
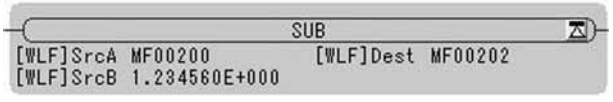
A B C

Item	Description	Usable Registers
A	Data output	• All integer type, double integer type, real number type registers (Excluding # and C registers) • The above registers with subscript • Subscript registers
B	Data input	• All integer type, double integer type, real number type registers (Excluding # and C registers) • The above registers with subscript • Subscript registers • Constants
C	Data to subtract	

(3) Programming Examples

Subtract (–) command programming examples are shown below.

◀ EXAMPLE ▶

Type	Motion Programs/ Sequence Programs	Ladder Programs
B	–	–
W	MW00101=MW00100-12345;	
L	ML00106=ML00102-ML00104;	
F	MF00202=MF00200-1.23456;	

8.5.4 Multiply (*)

Motion programs	Sequence Programs
Applicable	Applicable

(1) Overview

Multiply (*) performs integer and real number multiplication on the right side and stores the result in the register on the left side. With mixed integers and real numbers, the data type on the left side is stored.

(2) Format

MW00101 = MW00100 * 12345 ;
A B C

Item	Description	Usable Registers
A	Data output	• All integer type, double integer type, real number type registers (Excluding # and C registers) • The above registers with subscript • Subscript registers
B	Data input	• All integer type, double integer type, real number type registers (Excluding # and C registers) • The above registers with subscript • Subscript registers • Constants
C	Data to multiply	

(3) Programming Examples

Multiply (*) command programming examples are shown below.

◀ EXAMPLE ▶

Type	Motion Programs/ Sequence Programs	Ladder Programs
B	–	–
W	MW00102=MW00100*MW00101	
L	ML00106=ML00102*ML00104;	
F	MF00202=MF00200*1.23456;	

8.5.5 Divide (/)

Motion Programs	Sequence Programs
Applicable	Applicable

(1) Overview

Divide (/) performs integer and real number division on the right side and stores the result in the register on the left side. With mixed integers and real numbers, the data type on the left side is stored.

(2) Format

MW00101 = MW00100 / 12345 ;

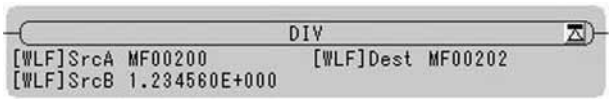
A B C

Item	Description	Usable Registers
A	Data output	• All integer type, double integer type, real number type registers (Excluding # and C registers) • The above registers with subscript • Subscript registers
B	Data input	• All integer type, double integer type, real number type registers (Excluding # and C registers) • The above registers with subscript • Subscript registers • Constants
C	Data to divide	

(3) Programming Examples

Divide (/) command programming examples are shown below.

◀ EXAMPLE ▶

Type	Motion Programs/ Sequence Programs	Ladder Programs
B	–	–
W	MW00102=MW00100/MW00101;	
L	ML00106=ML00102/ML00104;	
F	MF00202=MF00200/1.23456;	

8.5.6 Remainder (MOD)

Motion Programs	Sequence Programs
Applicable	Applicable

(1) Overview

When specified in the next block after Divide, MOD stores the remainder of the division in the specified variable. The remainder is stored as the data type on the left side.

(2) Format

```
MW00001 = 1000 / 999;
MW00002 = MOD;
```

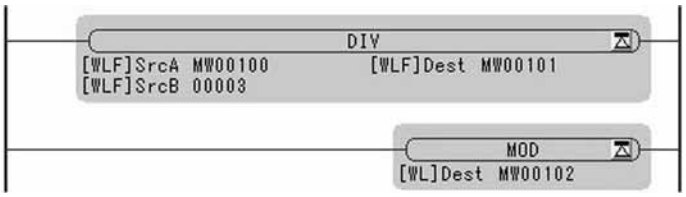
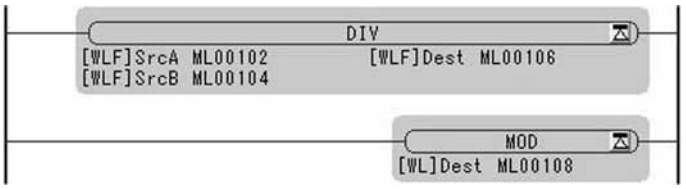
A

Item	Description	Usable Registers
A	Data output	• All integer and double integer type registers (Excluding # and C registers) • The above registers with subscript • Subscript registers

(3) Programming Examples

MOD command programming examples are shown below.

◀ EXAMPLE ▶

Type	Motion Programs/ Sequence Programs	Ladder Programs
B	–	–
W	MW00101=MW00100/3; MW00102=MOD;	
L	ML00106=ML00102/ML00104; ML00108=MOD;	
F	–	–

Example: Double integers

```
ML00106=ML00100*ML00102/ML00104;
(173575)  (100000)  (60000)  (34567)
ML00108=MOD;
(32975)
```

IMPORTANT

The MOD command must be specified in the next block after Divide. If it is not executed in the next block after Divide, the operation result cannot be guaranteed.

8.6 Logic Operation

This section explains the commands used to perform bit and integer logic operations.

Although operations that combine math operations are also possible, real number operations cannot be performed.

For priority levels of arithmetic operations, refer to *7.4 Priority Levels of Operations*.

8.6.1 OR (|)

Motion Programs	Sequence Programs
Applicable	Applicable

(1) Overview

OR performs a logical OR for the immediately preceding operation result and the specified registers, and returns the operation result. Real number registers cannot be used.

Table 8.3 Logical OR Truth Table (A=B|C)

B	C	A
0	0	0
0	1	1
1	0	1
1	1	1

(2) Format

MW00100 = DW00102 | AAAAH;

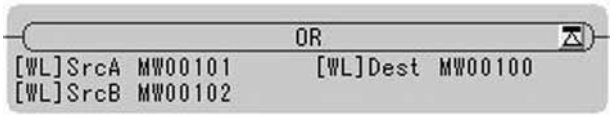
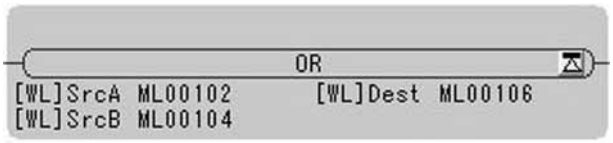
A B C

Item	Description	Usable Registers
A	Data output	• All bit type, integer type, and double integer type registers (Excluding # and C registers) • The above registers with subscript • Subscript registers
B, C	Data input	• All bit type, integer type, and double integer type registers (Excluding # and C registers) • The above registers with subscript • Subscript registers • Constants

(3) Programming Examples

Logical OR(|) command programming examples are shown below.

◀ EXAMPLE ▶

Type	Motion Programs/ Sequence Programs	Ladder Programs
B	MB001000=MB001010|MB001011;	
W	MW00100=MW00101|MW00102	
L	ML00106=ML00102|ML00104;	
F	–	–

8.6.2 AND (&)

Motion Programs	Sequence Programs
Applicable	Applicable

(1) Overview

AND (&) performs a logical AND for the immediately preceding operation result and the specified registers, and then returns the operation result. Real number registers cannot be used.

Table 8.4 Logical AND Truth Table (A=B&C)

B	C	A
0	0	0
0	1	0
1	0	0
1	1	1

(2) Format

MW00100 = DW00102 & AAAAH;

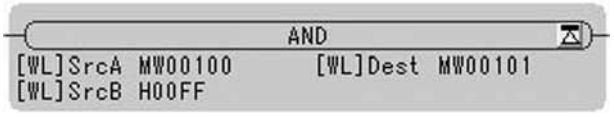
A B C

Item	Description	Usable Registers
A	Data output	• All bit type, integer type, and double integer type registers (Excluding # and C registers) • The above registers with subscript • Subscript registers
B, C	Data input	• All bit type, integer type, and double integer type registers (Excluding # and C registers) • The above registers with subscript • Subscript registers • Constants

(3) Programming Examples

The AND (&) command programming examples are shown below.

◀ EXAMPLE ▶

Type	Motion Programs/ Sequence Programs	Ladder Programs
B	MB001000=MB001010&MB001011;	
W	MW00101=MW00100&00FFH;	
L	ML00106=ML00102&ML00104;	
F	–	–

8.6.3 XOR (^)

Motion Programs	Sequence Programs
Applicable	Applicable

(1) Overview

XOR (^) performs an exclusive logical OR for the immediately preceding operation result and the specified registers, and then returns the operation result. Real number registers cannot be used.

Table 8.5 Exclusive Logical OR Truth Table
(A= B ^ C)

B	C	A
0	0	0
0	1	1
1	0	1
1	1	0

(2) Format

MW00100 = DW00102 ^ AAAAH;

A B C

Item	Description	Usable Registers
A	Data output	• All integer and double integer type registers (Excluding # and C registers) • The above registers with subscript • Subscript registers
BÅCC	Data input	• All integer and double integer type registers (Excluding # and C registers) • The above registers with subscript • Subscript registers • Constants

(3) Programming Examples

The XOR (^) command programming examples are shown below.

◀ EXAMPLE ▶

Type	Motion Programs/Sequence Programs	Ladder Programs
B	–	–
W	MW00101=MW00100 ^ 00FFH;	
L	ML00106=ML00102 ^ ML00104;	
F	–	–

8.6.4 NOT (!)

Motion Programs	Sequence Programs
Applicable	Applicable

(1) Overview

NOT inverts the data in the specified register and returns the operation result. Real number registers cannot be used.

(2) Format

MB001000 = ! MB001010;

A B

Item	Description	Usable Registers
A	Data output	• All bit type, integer type, and double integer type registers (Excluding # and C registers) • The above registers with subscript • Subscript registers
B	Data input	• All bit type, integer type, and double integer type registers (Excluding # and C registers) • The above registers with subscript • Subscript registers • Constants*

* Bit constants cannot be specified.

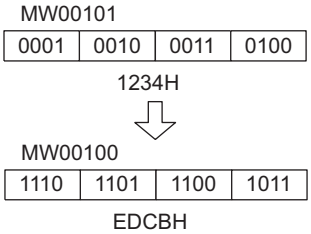
(3) Programming Examples

Logical NOT (!) programming examples are shown below.

◀ EXAMPLE ▶

Type	Motion Programs/ Sequence Programs	Ladder Programs
B	MB001000=!MB001010;	
W	MW00100=!MW00101;	
L	ML00100=!ML00102	
F	–	–

Example: MW00100=!MW00101;



8.7 Data Comparisons

This section explains the data comparison commands that are used for conditional expressions.

8.7.1 Data Comparison Commands (==, <>, >, <, >=, <=)

Motion Programs	Sequence Programs
Applicable	Applicable

(1) Overview

These commands are used to determine conditional expressions for commands such as branching commands (IF), repeat commands (WHILE), and I/O WAIT (IOW).

The following six comparison commands are provided.

Comparison Command	Meaning
= =	Match
< >	Mismatch
>	Greater than
<	Less than
> =	Greater than or equal to
< =	Less than or equal to

(2) Format

IF MB001000 == 1;

A

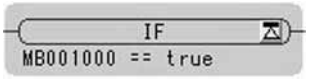
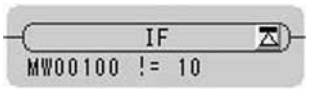
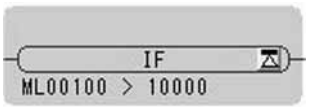
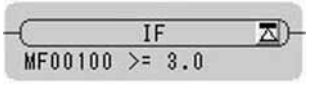
Item	Description	Usable Registers
A	Conditional expression	• All bit type*, integer type, double integer type, and real number type registers (Excluding # and C registers) • The above registers with subscript • Subscript registers

* Only the MATCH (= =) command can be used in a bit conditional expression.

(3) Programming Examples

Data comparison command programming examples are shown below.

◀ EXAMPLE ▶

Type	Motion Programs/ Sequence Programs	Ladder Programs
B	IF MB001000==1;	
W	IF MW00100<>10;	
L	IF ML00100>10000;	
F	IF MF00100>=3.0;	

The conditional expressions that can be used with data comparison commands are as shown below.

(a) Bit Data Comparison

Format	• Use == (MATCH) command for data comparison. • Specify a register on the left, and 0 or 1 on the right. IF MB000000 == 0; "MB000000 = 0 IF MB000000 == 1; "MB000000 = 1
Operations in Conditional Expression	• &, |, ! (AND, OR, NOT) can be used. IF (MB000000 & MB000001) == 1; "MB000000=1 AND MB000001=1 IF (MB000000 & !MB000001) == 1; "MB000000=1 AND MB000001=0 IF (MB000000 | MB000001) == 1; "MB000000=1 OR MB000001=1 IF (MB000000 | !MB000001) == 1; "MB000000=1 OR MB000001=0
Syntax Error Examples	• When <> (MISMATCH) is used: IF MB000000 <> 0; => Syntax error • When a numerical value is specified on the left, and a register on the right: IF 1 == MB000000; => Syntax error IF MB000000 == MB000001; => Syntax error • No data comparison command: IF MB000000; => Syntax error IF (0); => Syntax error • When more than one data comparison command is used: IF (MB000000 == 0) & (MB000001 == 1); => Syntax error

(b) Integer/Double Integer/Real Number Data Comparison

Format	• All data comparison commands (==, <>, >, <, >=, <=) can be used. • Specify a register either on the left or right. IF MW00000 == 3; "MW00000 = 3 IF ML00000 <> ML00002; "ML00000 ≠ ML00002 IF 1.23456 >= MF00000; "1.23456 ≥ MF00000
Operations in Conditional Expression	• Arithmetic operations and logic operations can be used. IF MW00000 == (MW00001/3); "MW00000 = (MW00001 ÷ 3) IF (ML00000 & F0000000H) <> ML00002; "(ML00000 ∧ F0000000H) ≠ ML00002 IF 1.23456 >= (MF00000 * MF00002); "1.23456 ≥ (MF00000 × MF00002)
Syntax Error Examples	• When a constant is specified both on the left and right: IF 0 == 3; => Syntax error IF (3.14*2*1000) > 9000.0; => Syntax error • No data comparison command: IF MW000000; => Syntax error IF (-1); => Syntax error • When more than one data comparison command is used: IF (MW00000 < 0) & (MW000001 > 0); => Syntax error

8.8 Data Operations

This section describes the data operation commands that are used to shift, transfer, and initialize data.

8.8.1 Bit Right Shift (SFR)

Motion Programs	Sequence Programs
Applicable	Applicable

(1) Overview

The SFR command shifts a bit string designated by the leading bit number and bit width the specified number of shifts to the right.

(2) Format

SFR MB001000 N5 W10 ;
A B C

Item	Description	Usable Registers
A	Leading bit	• All bit type registers (Excluding # and C registers) • The above registers with subscript • Subscript registers
B	Number of bits to be shifted	• All integer type registers (Excluding # and C registers) • The above registers with subscript • Subscript registers • Constants
C	Bit width	

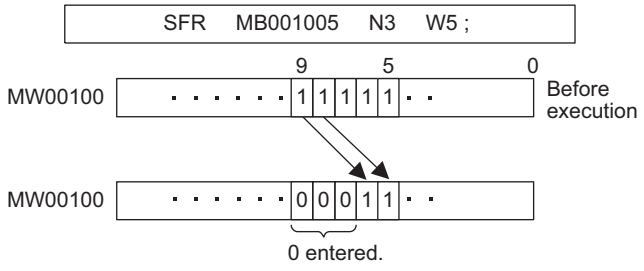
(3) Programming Examples

SFR command programming examples are shown below.

◀ EXAMPLE ▶

Type	Motion Programs/ Sequence Programs	Ladder Programs
B	–	–
W	SFR MB001000 N5 W10;	
L	–	–
F	–	–

Example: Five bits with MB001005 (bit 5 of MW00100) as the leading bit are shifted three bits to the right.



INFO

With the SFR command, if the number of shifts is greater than the bit width, all data with the specified bit width will be set to 0.

8.8.2 Bit Left Shift (SFL)

Motion Programs	Sequence Programs
Applicable	Applicable

(1) Overview

The SFL command shifts a bit string designated by the leading bit number and bit width the specified number of shifts to the left.

(2) Format

SFL MB001000 N5 W10 ;
A B C

Item	Description	Usable Registers
A	Leading bit	• All bit type registers (Excluding # and C registers) • The above registers with subscript • Subscript registers
B	Number of bits to be shifted	• All integer type registers (Excluding # and C registers) • The above registers with subscript • Subscript registers • Constants
C	Bit width	

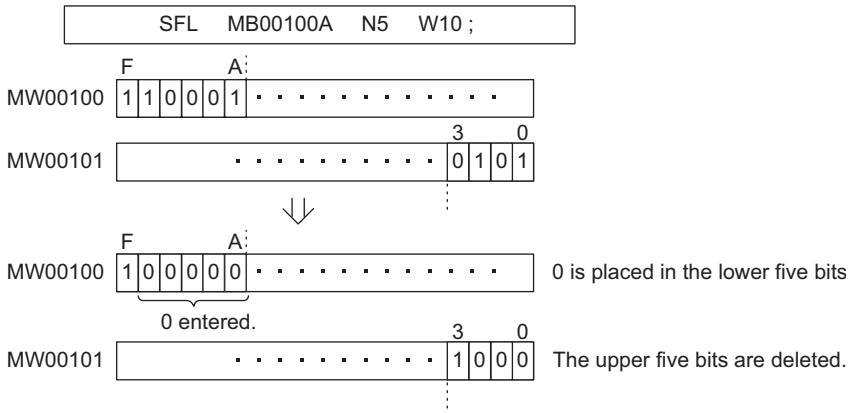
(3) Programming Examples

SFL command programming examples are shown below.

◀ EXAMPLE ▶

Type	Motion Programs/ Sequence Programs	Ladder Programs
B	–	–
W	SFL MB001000 N5 W10;	
L	–	–
F	–	–

Examples: Ten bits with MB00100A (bit A of MW00100) as the leading bit are shifted five bits to the left.



INFO

With the SFL command, if the number of shifts is greater than the bit width, all data with the specified bit width will be set to 0.

8.8.3 Block Move (BLK)

Motion Programs	Sequence Programs
Applicable	Applicable

(1) Overview

The BLK command moves the specified number of words from the beginning of the source register to the beginning of the destination register.

(2) Format

BLK MW00100 DW00100 W10 ;
A B C

Item	Description	Usable Registers
A	Source leading register	• All integer type registers (Excluding # and C registers) • The above registers with subscript • Subscript registers
B	Destination leading register	• All integer type registers (Excluding # and C registers) • The above registers with subscript • Subscript registers
C	Number of blocks to be moved	• Constants

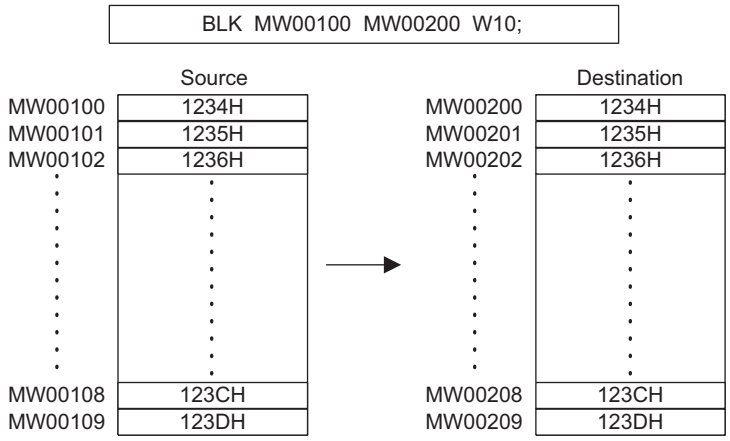
(3) Programming Examples

BLK command programming examples are shown below.

◀ EXAMPLE ▶

Type	Motion Programs/ Sequence Programs	Ladder Programs
B	–	–
W	BLK MW00100 DW00100 W10;	
L	–	–
F	–	–

Example: MW00100 to MW00109 are moved to MW00200 to MW00209.



INFO As long as the source registers and destination registers are not overlapped, the source data is moved to the destination registers as it is. If overlapped, the source data may not be moved to the destination registers as it is.

8.8.4 Clear (CLR)

Motion Programs	Sequence Programs
Applicable	Applicable

(1) Overview

The CLR command clears the specified number of blocks from the leading data clear register, i.e., sets it to 0.

(2) Format

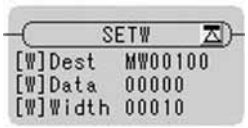
CLR MW00100 W10 ;
A B

Item	Description	Usable Registers
A	Leading register whose data is cleared	• All integer type registers (Excluding # and C registers) • The above registers with subscript • Subscript registers
B	Number of blocks	• All integer type registers (Excluding # and C registers) • The above registers with subscript • Subscript registers • Constants

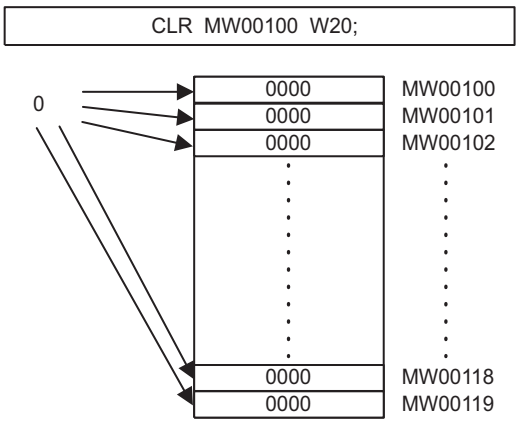
(3) Programming Examples

CLR command programming examples are shown below.

◀ EXAMPLE ▶

Type	Motion Programs/ Sequence Programs	Ladder Programs
B	–	–
W	CLR MW00100 W10;	
L	–	–
F	–	–

Example: The data of registers from MW00100 to MW00119 are cleared to 0.



8.8.5 ASCII Conversion 1 (ASCII)

Motion Programs	Sequence Programs
Applicable	Applicable

(1) Overview

The ASCII command converts the character string specified during command execution into ASCII code and stores it in the specified register (integer register). Upper and lower case letters can be distinguished from each other.

The first character and the second character are stored respectively in the lower byte and upper byte of the first word in order. If the number of characters in the string is odd, the upper byte of the last word in the storage destination register becomes 0. The number of input characters is up to 32.

Note: The following versions of system software and programming tool MPE720 are required to use an ASCII command.

MP2000 Series Machine Controller	Applicable Version Numbers
All models	Ver2.60 or later

MPE720	Applicable Version Numbers
MPE720 Ver.5	MPE720 Ver.5.38 or later
MPE720 Ver.6	MPE720 Ver.6.04 or later MPE720 Ver.6.04Lite or later

(2) Format

ASCII ‘ABCDEFG’ MW00200;
A B

Item	Description	Usable Registers
A	Character string	ASCII characters
B	Storage register number	Integer type registers (Excluding # and C registers)

The following tables show the characters that can be used in the ASCII command.

(a) Usable Characters

Item	ASCII Characters
Alphanumeric Characters	a to z, A to Z, 0 to 9
Symbols	Space, ! # $ % & ( ) * + , - . / : ; < = > ? @ [ ] //] ^ _ ` { | } ~

(b) Usable Characters

Item	ASCII Characters
Single Quotation	’
Double Quotation	"
Double Slash	//

(3) Programming Examples

ASCII command programming examples are shown below.

(a) Storing the character string “ABCD” in MW00100 to MW00101

◀ EXAMPLE ▶

```
ASCII 'ABCD'  MW00100;
```

	Upper Byte	Lower Byte	
MW00100	42H('B')	41H('A')	MW00100 = 4241H
MW00101	44H('D')	43H('C')	MW00101 = 4443H

(b) Storing the character string “ABCDEFG” in MW00100 to MW00103

◀ EXAMPLE ▶

```
ASCII 'ABCDEFG'  MW00100;
```

	Upper Byte	Lower Byte	
MW00100	42H('B')	41H('A')	MW00100 = 4241H
MW00101	44H('D')	43H('C')	MW00101 = 4443H
MW00102	46H('F')	45H('E')	MW00102 = 4645H
MW00103	00H	47H('G')	MW00103 = 0047H

↑ 0 will be entered in the remaining byte.

8.9 Basic Functions

This section describes the basic function commands, including trigonometric functions, square roots, binary data conversion, and BCD data conversion.

8.9.1 Sine (SIN)

Motion Programs	Sequence Programs
Applicable	Applicable

(1) Overview

The Sine command (SIN) returns the sine of integer or real number data as the operation result. Double integer data cannot be used.

(2) Format

MW00100 = SIN (3000);
A B

Item	Description	Unit	Usable Registers
A	Sine value output	–	• All integer and real number type registers (Excluding # and C registers) • The above registers with subscript • Subscript registers
B	Angle input	Degree (°)*	• All integer and real number type registers (Excluding # and C registers) • The above registers with subscript • Subscript registers • Constants

* The input units and output results are different for integer and real number data.

- Integer Data
  Integer data can be used within a range of -327.68 to 327.67 degrees. The immediately preceding operation result (integer data) is used as the input, and the operation result is returned in an integer register (input unit1 = 0.01 degrees). The operation result is multiplied by 10000 before being output.
- Real Number Data
  The command will uses the immediately preceding operation result (real number data) as input, and return the sine in a real number register (unit = degrees).

Example:

Integer Data	Real Number Data
MW00102 = SIN (MW00100) ; (05000) (03000) Equivalent => 0.5=SIN30°	MF00102 = SIN (MF00100) ; (0.5) (30.0)

IMPORTANT

If integer data is input outside the range of -327.68 to 327.67 degrees, a correct result cannot be obtained.

(3) Programming Examples

SIN command programming examples are shown below.

◀ EXAMPLE ▶

Type	Motion Programs/ Sequence Programs	Ladder Programs
B	–	–
W	MW00102=SIN(MW00100);	
L	–	–
F	DF00202=SIN(DF00200);	

8.9.2 Cosine (COS)

Motion Programs	Sequence Programs
Applicable	Applicable

(1) Overview

The cosine command (COS) returns the cosine of integer or real number data as the operation result. Double integer data cannot be used.

(2) Format

MW00100 = COS (3000);

A B

Item	Description	Unit	Usable Registers
A	Cosine value output	–	• All integer and real number type registers (Excluding # and C registers) • The above registers with subscript • Subscript registers
B	Angle input	Degree (°)*	• All integer and real number type registers (Excluding # and C registers) • The above registers with subscript • Subscript registers • Constants

* The input units and output results are different for integer and real number data.

- Integer Data
  Integer data can be used within a range of -327.68 to 327.67 degrees. The immediately preceding operation result (integer data) is used as input, and the operation result is returned in an integer register (input unit 1 = 0.01 degrees). The operation result is multiplied by 10000 before being output.
- Real Number Data
  The command uses the immediately preceding operation result (real number data) as input, and returns the cosine in a real number register (unit = degrees).

Example:

Integer Data	Real Number Data
MW00102 = COS (MW00100) ; (05000) (06000) Equivalent => 0.5=COS60°	MF00102 = COS (MF00100) ; (0.5) (60.0)

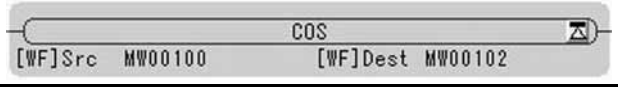
IMPORTANT

If integer data is input outside the range of -327.68 to 327.67 degrees, a correct result cannot be obtained.

(3) Programming Examples

COS command programming examples are shown below.

◀ EXAMPLE ▶

Type	Motion Programs/ Sequence Programs	Ladder Programs
B	–	–
W	MW00102=COS(MW00100);	
L	–	–
F	DF00202=COS(DF00200);	

8.9.3 Tangent (TAN)

Motion Programs	Sequence Programs
Applicable	Applicable

(1) Overview

The TAN command uses the specified variable or constant (unit = degrees) as input and returns the tangent in a real number register.

(2) Format

MW00100 = TAN (1.0);
A B

Item	Description	Unit	Usable Registers
A	Tangent value output	–	• All real number type registers (Excluding # and C registers) • The above registers with subscript • Subscript registers
B	Angle input	Degree (°)*	• All real number type registers (Excluding # and C registers) • The above registers with subscript • Subscript registers • Constants

* Example: Calculates the tangent of the input value (θ = 45.0°): TAN (θ) = 1.0

```
DF00102=TAN(DF00100);
(1.0)          (45.0)
```

IMPORTANT

The TAN command can use only real number data. If bits, integers, or double integers are specified, an error will result at compilation.

(3) Programming Examples

TAN command programming examples are shown below.

◀ EXAMPLE ▶

Type	Motion Programs/ Sequence Programs	Ladder Programs
B	–	–
W	–	–
L	–	–
F	DF00202=TAN(DF00200);	

8.9.4 Arc Sine (ASN)

Motion Programs	Sequence Programs
Applicable	Applicable

(1) Overview

The ASN command uses the specified variable or constant as input and returns the arc sine (unit = degrees) in a real number register.

(2) Format

```
MF00100 = ASN (0.5);
    A         B
```

Item	Description	Unit	Usable Registers
A	Angle output	Degree (°)*	• All real number type registers (Excluding # and C registers) • The above registers with subscript • Subscript registers
B	Sine value input	–	• All real number type registers (Excluding # and C registers) • The above registers with subscript • Subscript registers • Constants

* Example: Calculates the arc sine of the input value (0.5): ASN (0.5) = 30.0 degrees.

```
MF00202=ASN(MF00200);
 (30.0)       (0.5)
```

IMPORTANT

The ASN command can use only real number data. If bits, integers, or double integers are specified, an error will result at compilation.

(3) Programming Examples

ASN command programming examples are shown below.

◀ EXAMPLE ▶

Type	Motion Programs/ Sequence Programs	Ladder Programs
B	–	–
W	–	–
L	–	–
F	DF00202=ASN(DF00200);	

8.9.5 Arc Cosine (ACS)

Motion Programs	Sequence Programs
Applicable	Applicable

(1) Overview

The ACS command uses the specified variable or constant as input and returns the arc cosine (unit = degrees) in a real number register.

(2) Format

MF00100 = ACS (0.5);

A B

Item	Description	Unit	Usable Registers
A	Angle output	Degree (°)*	• All real number type registers (Excluding # and C registers) • The above registers with subscript • Subscript registers
B	Cosine value input	–	• All real number type registers (Excluding # and C registers) • The above registers with subscript • Subscript registers • Constants

* Example: Calculates the arc cosine of the input value (0.5): ACS (0.5) = 60.0 degrees.

MF00100 = ACS (MF00102) ;

(60.0) (0.5)

IMPORTANT

The ACS command can use only real number data. If bits, integers, or double integers are specified, an error will result at compilation.

(3) Programming Examples

ACS command programming examples are shown below.

◀ EXAMPLE ▶

Type	Motion Programs/ Sequence Programs	Ladder Programs
B	–	–
W	–	–
L	–	–
F	DF00202=ACS(DF00200);	

8.9.6 Arc Tangent (ATN)

Motion Programs	Sequence Programs
Applicable	Applicable

(1) Overview

The ATN command returns the arc tangent of integer or real number data as the operation result. Double integer data cannot be used.

(2) Format

MW00100 = ATN (100);
A B

Item	Description	Unit	Usable Registers
A	Angle output	Degree (°)*	• All integer and real number type registers (Excluding # and C registers) • The above registers with subscript • Subscript registers
B	Tangent value input	–	• All integer and real number type registers (Excluding # and C registers) • The above registers with subscript • Subscript registers • Constants

* The input units and output results are different for integer and real number data.

- Integer Data
  Integer data can be used within a range of -327.68 to 327.67 degrees. The immediately preceding operation result (integer data) is used as the input, and the operation result is returned in an integer register (input unit 1 = 0.01 degrees). The operation result is multiplied by 100 before being output.
- Real Number Data
  The command uses the immediately preceding operation result (real number data) as input, and returns the arc tangent in a real number register.

Example:

Integer Data	Real Number Data
MW00100 = ATN (MW00102) ; (04500) (00100) Equivalent => 45=ATN(1.0)	MF00100 = ATN (MF00102) ; (45.0) (1.0)

(3) Programming Examples

ATN command programming examples are shown below.

◀ EXAMPLE ▶

Type	Motion Programs/ Sequence Programs	Ladder Programs
B	–	–
W	MW00102=ATN(MW00100);	
L	–	–
F	DF00202=ATN(DF00200);	

8.9.7 Square Root (SQT)

Motion Programs	Sequence Programs
Applicable	Applicable

(1) Overview

The SQT command returns the square root of an integer or real number as the operation result. Double integer data cannot be used.

(2) Format

MW00100 = SQT (100);

A B

Item	Description	Usable Registers
A	Root value output	• All integer and real number type registers (Excluding # and C registers) • The above registers with subscript • Subscript registers
B	Data input	• All integer and real number type registers (Excluding # and C registers) • The above registers with subscript • Subscript registers • Constants

Note: The input units and output results are different for integer and real number data.

- Integer Data
  The result is different from that obtained for the mathematical square root, and is calculated using the following formula:

  $$32768 * \mathrm{sign}(B) * \sqrt{|B|/32768}$$

  sign(B): Sign for data input
  |B|: Absolute value of data input

  That is to say, the output is the result of the mathematically expressed square root multiplied by 32768. When the input is a negative number, an absolute square root is calculated, and the negative number is taken as the operation result. The operation error is a maximum of ±2.
- Real Number Data
  The SQT command uses the immediately preceding operation result (real number data) as input and returns the square root in a real number register.

Example:

Input Value	Integer Data	Real Number Data
Positive value input	MW00100 = SQT (MW00102) ; (01448) (00064) $\sqrt{64} \times \sqrt{32768} = 1448$ (8) (181)	MF00100 = SQT (MF00102) ; (8.0) (64.0)
Negative value input	MW00100 = SQT (MW00102) ; (-01448) (-00064) $-\left(\sqrt{64} \times \sqrt{32768}\right) = -1448$ (8) (181)	MF00100 = SQT (MF00102) ; (-8.0) (-64.0)

(3) Programming Examples

SQT command programming examples are shown below.

◀ EXAMPLE ▶

Type	Motion Programs/ Sequence Programs	Ladder Programs
B	–	–
W	MW00102=SQT(MW00100);	
L	–	–
F	DF00202=SQT(DF00200);	

8.9.8 BCD to Binary (BIN)

Motion Programs	Sequence Programs
Applicable	Applicable

(1) Overview

The BCD to Binary (BIN) command converts BCD data to binary data.

Only integer data can be used. If a non-BCD data is specified, a correct result cannot be obtained.

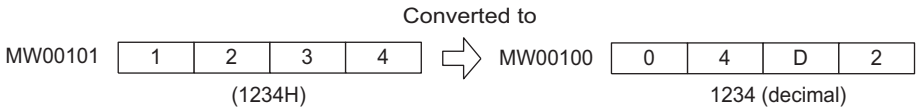
(2) Format

MW00100 = BIN (1234H);

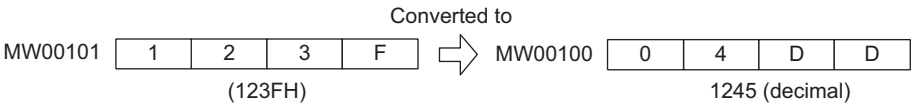
A B

Item	Description	Usable Registers
A	Binary data output	• All integer and double integer type registers (Excluding # and C registers) • The above registers with subscript • Subscript registers
B	BCD data input	• All integer and double integer type registers (Excluding # and C registers) • The above registers with subscript • Subscript registers • Constants

Note: Example 1



Example 2



INFO If non-BCD data is specified, a correct result cannot be obtained.

(3) Programming Examples

BIN command programming examples are shown below.

◀ EXAMPLE ▶

Type	Motion Programs/ Sequence Programs	Ladder Programs
B	–	–
W	MW00101=BIN(MW00100);	
L	ML00102=BIN(ML00100);	
F	–	–

8.9.9 Binary to BCD (BCD)

Motion Programs	Sequence Programs
Applicable	Applicable

(1) Overview

The Binary to BCD (BCD) command converts binary data to BCD data.

Only integer data can be used. If the binary data is 9999 or higher or is a negative value, a correct result cannot be obtained.

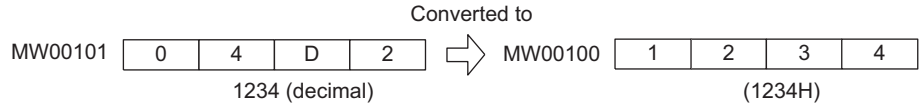
(2) Format

MW00100 = BCD (1234);

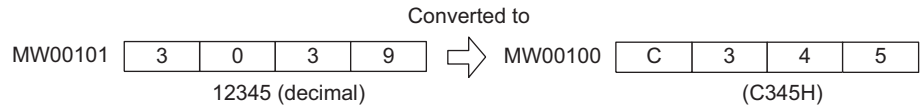
A B

Item	Description	Usable Registers
A	BCD data output	• All integer and double integer type registers (Excluding # and C registers) • The above registers with subscript • Subscript registers
B	Binary data input	• All integer and double integer type registers (Excluding # and C registers) • The above registers with subscript • Subscript registers • Constants

Note: Example 1



Example 2

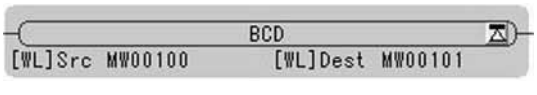
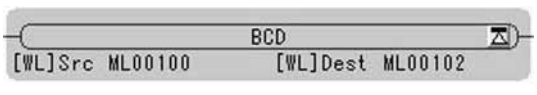


INFO If the binary data is greater than 9999, a correct result cannot be obtained.

(3) Programming Examples

BCD command programming examples are shown below.

◀ EXAMPLE ▶

Type	Motion Programs/ Sequence Programs	Ladder Programs
B	–	–
W	MW00101=BCD(MW00100);	
L	ML00102=BCD(ML00100);	
F	–	–

8.9.10 Set Bit (S{ })

Motion Programs	Sequence Programs
Applicable	Applicable

(1) Overview

This command turns ON the specified bit if the logical operation result is true. It does not turn OFF the specified bit, even if the logical operation result is false.

(2) Format

S { MB001000 } = MB001010 & MB001011;

A B

Item	Description	Usable Registers
A	Specified bit	• All bit type registers (Excluding # and C registers) • The above registers with subscript
B	Logic expression	• All bit type registers (Excluding # and C registers) • The above registers with subscript • Constants

(3) Programming Examples

The Set Bit command (S{ }) programming examples are shown below.

◀ EXAMPLE ▶

Type	Motion Programs/ Sequence Programs	Ladder Programs
B	S{MB001000}=MB001010& MB001011;	
W	–	–
L	–	–
F	–	–

8.9.11 Reset Bit (R{ })

Motion Programs	Sequence Programs
Applicable	Applicable

(1) Overview

This command turns OFF the specified bit if the logical operation result is true. It does not turn ON the specified bit, even if the logical operation result is false.

(2) Format

R { MB001000 } = MB001010 & MB001011;

A B

Item	Description	Usable Registers
A	Specified bit	• All bit type registers (Excluding # and C registers) • The above registers with subscript
B	Logic expression	• All bit type registers (Excluding # and C registers) • The above registers with subscript • Constants

(3) Programming Examples

Reset Bit command (R{ }) programming examples are shown below.

◀ EXAMPLE ▶

Type	Motion Programs/ Sequence Programs	Ladder Programs
B	R{MB001000}=MB001010& MB001011;	
W	–	–
L	–	–
F	–	–

8.9.12 Rising Pulse (PON)

Motion Programs	Sequence Programs
Not applicable	Applicable

(1) Overview

The PON command is ON during one scan of bit output when the bit input status changes from OFF to ON. The register that stores the previous bit output value is used as a work of PON processing. Set the registers that are not used for other processes.

Note: The following versions of system software and programming tool MPE720 are required to use PON commands.

MP2000 Series Machine Controller	Required Version
All models	Ver.2.60 or later

MPE720	Required Version
MPE720 Ver.5	MPE720 Ver.5.38 or later
MPE720 Ver.6	MPE720 Ver.6.04 or later MPE720 Ver.6.04 Lite or later

(2) Format

DB000002 = PON (DB000000 DB000001) ;

A B C

Item	Description	Usable Registers
A	Bit output	• Bit type registers (Excluding # and C registers) • The above registers with subscript
B	Bit input	• All bit type registers • The above registers with subscript
C	To store the previous bit output value	• Bit type registers (Excluding # and C registers) • The above registers with subscript

(3) Programming Examples

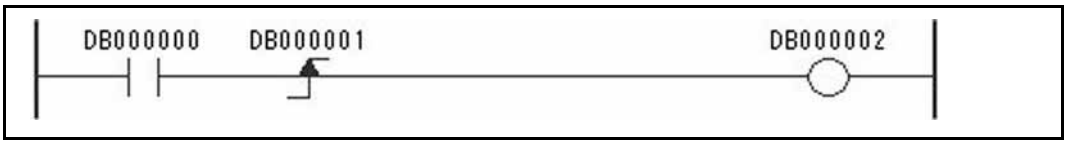
The PON command programming examples are shown below.

(a) Outputting to a Coil

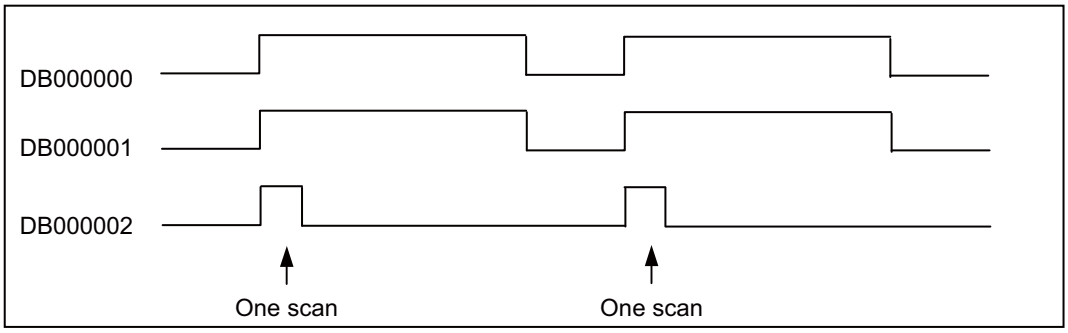
◀ EXAMPLE ▶

```
DB000002=PON(DB000000   DB000001);
```

- Ladder equivalent circuit



- Timing chart

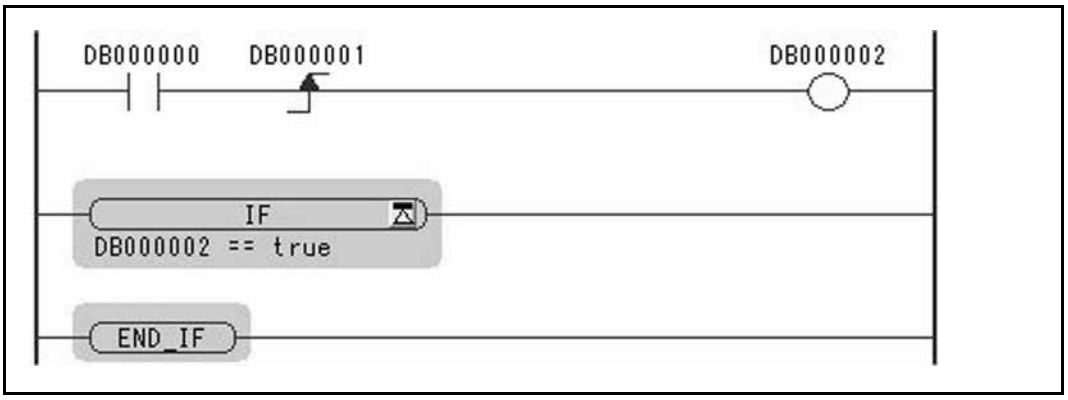


(b) Using a PON Command Combined with an IF Command

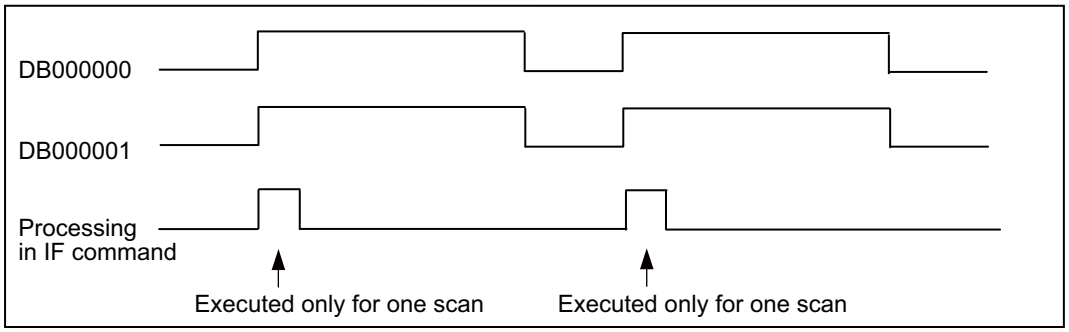
◀ EXAMPLE ▶

```
IF PON(DB000000   DB000001) == 1;
          •
          •
IEND;
```

- Ladder equivalent circuit



- Timing chart



8.9.13 Falling Pulse (NON)

Motion Programs	Sequence Programs
Not applicable	Applicable

(1) Overview

Falling Pulse (NON) is ON during one scan of bit output when the bit input status changes from ON to OFF. The register that stores the previous bit output value is used as a work of NON processing. Set the registers that are not used for other processes.

Note: The following versions of system software and programming tool MPE720 are required to use NON commands.

MP2000 Series Machine Controller	Required Version
All models	Ver.2.60 or later

MPE720	Required Version
MPE720 Ver.5	MPE720 Ver.5.38 or later
MPE720 Ver.6	MPE720 Ver.6.04 or later MPE720 Ver.6.04 Lite or later

(2) Format

DB000002 = NON (DB000000 DB000001) ;

A B C

Item	Description	Usable Registers
A	Bit output	• Bit type registers (Excluding # and C registers) • The above registers with subscript
B	Bit input	• All bit type registers • The above registers with subscript
C	To store the previous bit output value	• Bit type registers (Excluding # and C registers) • The above registers with subscript

(3) Programming Examples

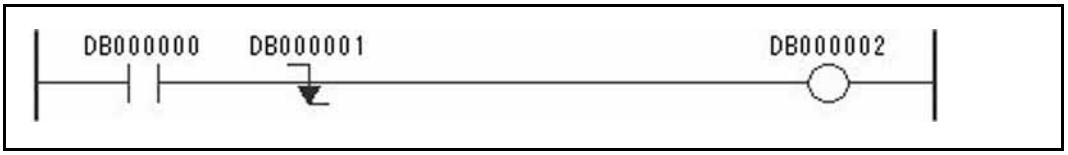
NON command programming examples are shown below.

(a) Outputting to a Coil

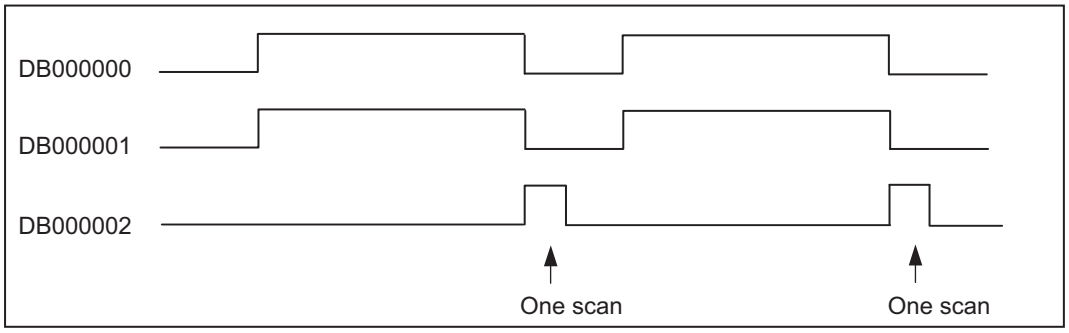
```
DB000002=NON(DB000000   DB000001);
```

- Ladder equivalent circuit

◀ EXAMPLE ▶



- Timing chart

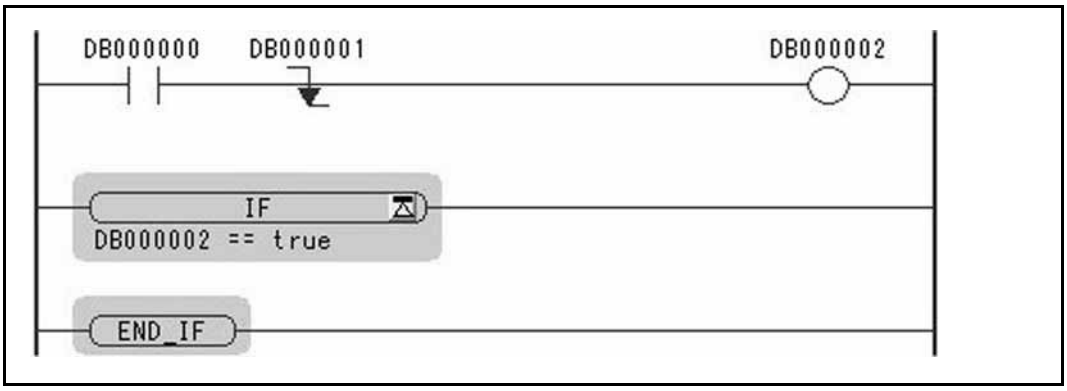


(b) Using a NON Command Combined with an IF Command

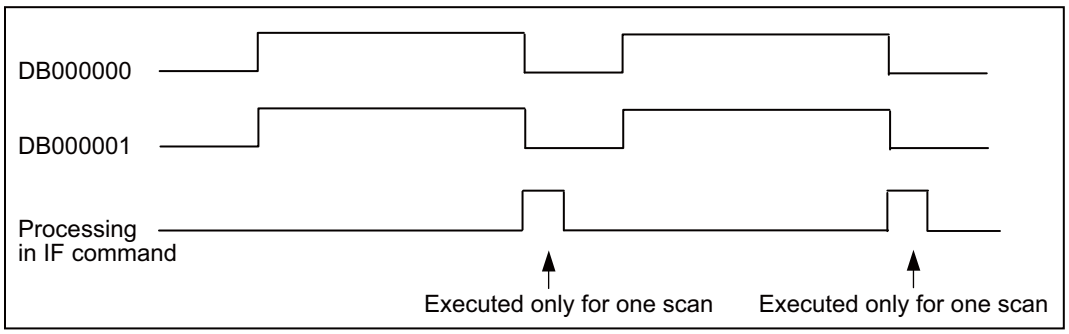
◀ EXAMPLE ▶

```
IF NON(DB000000   DB000001) == 1;
        ÅE
        ÅE
IEND;
```

- Ladder equivalent circuit



- Timing chart



8.9.14 ON-Delay Timer (TON): Counting unit: 0.01 second

Motion Programs	Sequence Programs
Not applicable	Applicable

(1) Overview

The TON command counts the milliseconds (in units of 10 ms) when the bit input is ON. When the counted value is equal to the set value, the bit output will turn ON. If the bit input turns OFF during counting, the timer operation will stop. After the bit input turns ON again, counting starts again from the beginning (0). In the registers for counting, the actual count (in units of 10 ms) is stored.

Note: The following versions of system software and programming tool MPE720 are required to use TON commands.

MP2000 Series Machine Controller	Required Version
All models	Ver.2.60 or later

MPE720	Required Version
MPE720 Ver.5	MPE720 Ver.5.38 or later
MPE720 Ver.6	MPE720 Ver.6.04 or later MPE720 Ver.6.04Lite or later

(2) Format

```
DB000001 = DB000000 & TON (500  DW00001);
   A        B          C     D
```

Item	Description	Usable Registers
A	Bit output	• Bit type registers (Excluding # and C registers) • The above registers with subscript
B	Bit input	• All bit type registers • The above registers with subscript
C	Set value	• All integer type registers • The above registers with subscript • Constants (0 to 65535 (655.35 s): Every 10 ms)
D	Register for timer counting	• All integer type registers • The above registers with subscript

IMPORTANT

- Milliseconds are not counted while the debugging operation is stopped.
  Counting starts again from the current counted value after the debugging operation restarts.
- Be sure to designate bit input “DBxxxxxx&.”

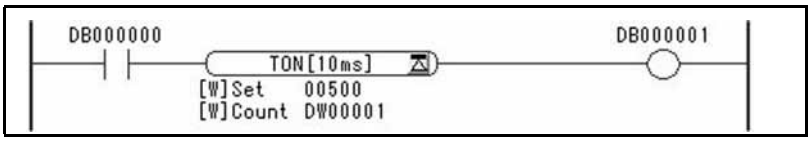
(3) Programming Examples

TON command programming examples are shown below.

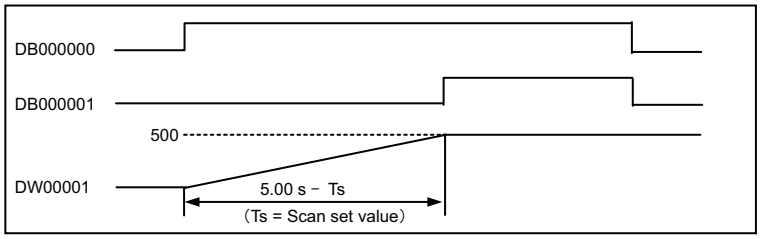
◀ EXAMPLE ▶

```
DB000001=DB000000 & TON (500   DW00001);
                          ↑ Set to 5 seconds.
```

- Ladder equivalent circuit



- Timing chart



8.9.15 OFF-Delay Timer (TOF):Counting unit: 0.01 second

Motion Programs	Sequence Programs
Not applicable	Applicable

(1) Overview

The TOF command counts the milliseconds (in units of 10 ms) when the bit input is OFF. When the counted value is equal to the set value, the bit output will turn OFF. If the bit input turns ON during counting, the timer operation will stop. After the bit input turns OFF again, counting starts again from the beginning (0). In the registers for counting, the actual count (in units of 10 ms) is stored.

Note: The following versions of system software and programming tool MPE720 are required to use TOF commands.

MP2000 Series Machine Controller	Required Version
All models	Ver. 2.60 or later

MPE720	Required Version
MPE720 Ver.5	MPE720 Ver.5.38 or later
MPE720 Ver.6	MPE720 Ver.6.04 or later MPE720 Ver.6.04Lite or later

(2) Format

DB000001 = DB000000 & TOF (500 DW00001);

A B C D

Item	Description	Usable Registers
A	Bit output	• Bit type registers (Excluding # and C registers) • The above registers with subscript
B	Bit input	• All bit type registers • The above registers with subscript
C	Set value	• All integer type registers • The above registers with subscript • Constants (0 to 65535 (655.35 s): Every 10 ms)
D	Register for timer counting	• All integer type registers • The above registers with subscript

IMPORTANT

- Milliseconds are not counted while the debugging operation is stopped.
  Counting starts again from the current counted value after the debugging operation restarts.
- Be sure to designate bit input “DBxxxxxx&.”

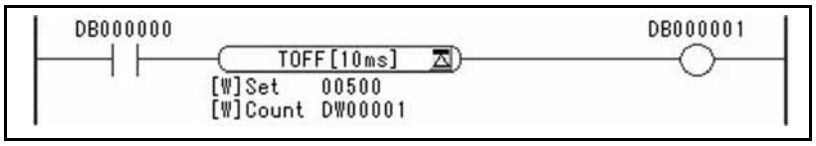
(3) Programming Examples

TOF command programming examples are shown below.

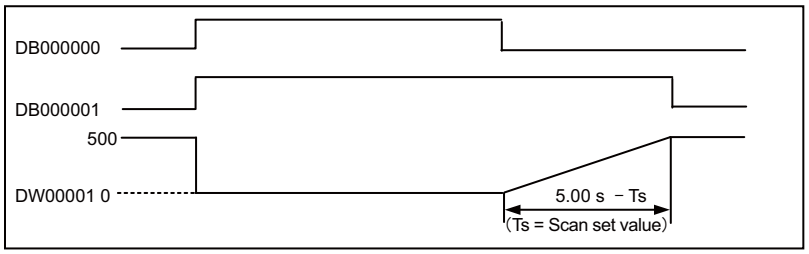
◀ EXAMPLE ▶

```
DB000001=DB000000 & TOF (500   DW00001);
```

- Ladder equivalent circuit



- Timing chart



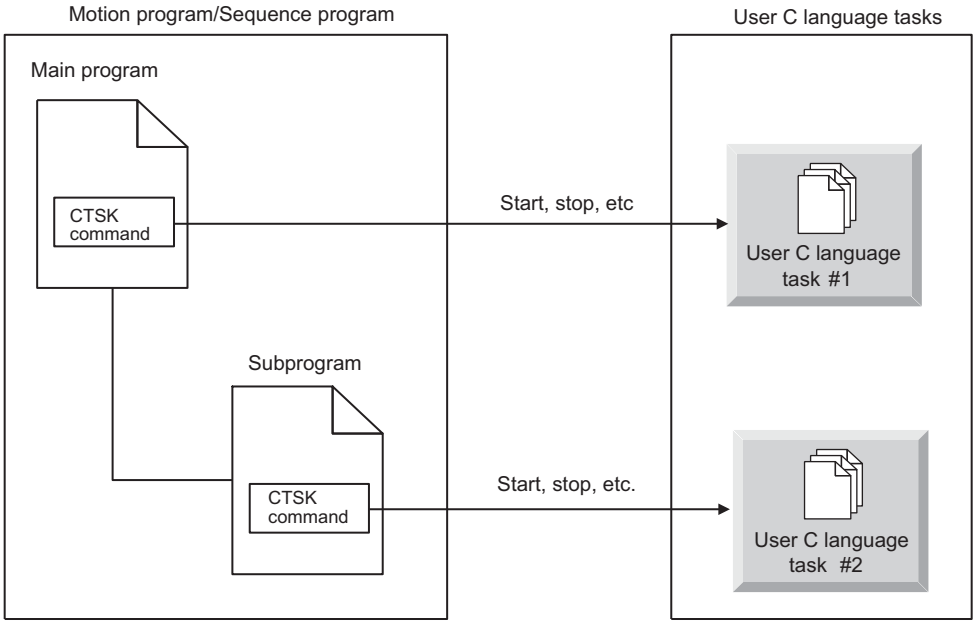
8.10 C-Language Control Commands

8.10.1 C-Language Task Control (CTSK)

Motion Programs	Sequence Programs
Applicable	Applicable

(1) Overview

The C-Language Task Control command (CTSK) is used to control operations, including start and stop, of user C-language tasks.



Note: The following versions of system software and programming tool MPE720 are required to use CTSK commands.

MP2000 Series Machine Controller	Required Version
All models	Ver. 2.60 or later

MPE720	Required Version
MPE720 Ver.5	MPE720 Ver.5.38 or later
MPE720 Ver.6	MPE720 Ver.6.04 or later MPE720 Ver.6.04Lite or later

INFO

The Machine Controller MP2000 series embedded C-language package is required to use user C-language tasks. For details, refer to *Machine Controller MP2000 Series Embedded C-Language Programming Package Development Guide* (manual no.: SIEP C880700 25).

(2) Format

```
CTSK EXECUTE TYPE, C_NAME, COMPLETE ERROR ERR_CODE;
```

I/O Definition	No.	Name	I/O Designation	Description	
Input	1	EXECUTE	B-VAL	Designates of execution of CTSK function.	
	2	TYPE	I-REG	Designates the task control type.	
				1: WAKEUP	Wakes up the task from the WAIT status.
				2: RESET	Valid only for sequence programs (low scan) Exits and deletes the task once, and then creates and starts a task. The started task then enters the WAIT status.
				3: SUSPEND	Interrupts the task and moves the task to the SUSPEND status.
				4: RESUME	Moves the task from the SUSPEND status to the READY status.
	3	C_NAME	Address input	Designates the leading register number (MW/DW address) of the registers where the user C-language task name (project name) is stored.	
Output	1	COMPLETE	B-VAL	Completes execution of CTSK function	
	2	ERROR	B-VAL	Error occurrence (The contents of error is reported to ERR_CODE.)	
	3	ERR_CODE	L-REG	Error code	
				0x00000000	No error
				0x0000006F	DWG (drawing) type error • CTSK function was executed by a sequence program (start drawing)
				0x00000091	• TYPE setting error • TYPE set value is out of the range. • The task was neither in the WAIT nor WAIT-SUSPEND status when the task control type WAKEUP was executed. • The task was neither in the WAIT nor READY status when the task control type SUSPEND was executed.
				0x00000094	The task designated in C_NAME does not exist.
				0x00000096	C_NAME register over the upper/lower limit Note: This can be detected also when EXECUTE input is OFF.
				0xFFFFFFDD	µITRON detection error (Invalid ID number)*
				0xFFFFFFCC	µITRON detection error (Task not registered)*
				0xFFFFFFC1	µITRON detection error (Illegal object status)* • The task is in DORMANT status. • RESUME was commanded to a task that was not in SUSPEND status.
				0xFFFFFFBB	µITRON detection error (Context eror)* • Cannot issue from the task independent context.
				0xFFFFFFB7	µITRON detection error (Queue overflow)*

* The µITRON detection errors will not occur normally because the system manages them.

Note: 1. The EXECUTE input is not used for the signal rising edge, but treated as level. This is to implement task control in every scan cycle.

2. The task controls except for RESET can be executed by motion programs, sequence programs for high-speed scan drawing, and sequence programs for low-speed scan drawing. They cannot be executed by sequence programs for start drawing.

3. The task control type RESET can be executed only by sequence programs for low-speed scan drawing.

(3) Programming Examples

A CTSK command programming example is shown below.

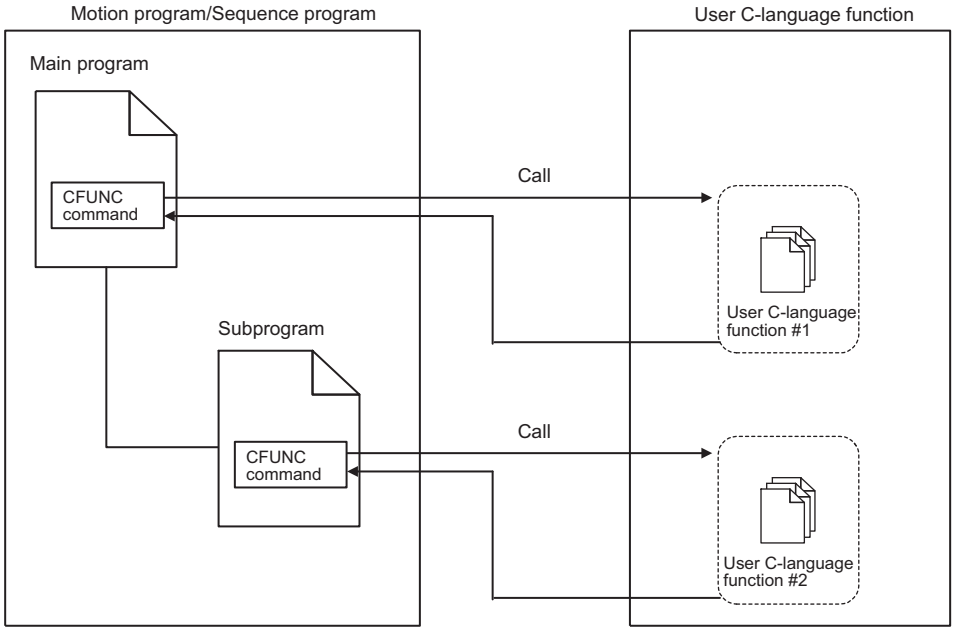
◀ EXAMPLE ▶

```
ASCII 'ctask1' DW00010;      "User C-language task name
DW00013 = 0000H;             "NULL code
DW00002 = 1;                 "WAKEUP
DB000001 = 1;                "Execute the task control
CTSK DB000001 DW00002, DA00010, DB000002 DB000003 DL00004;
```

8.10.2 C-Language Function Call (CFUNC)

(1) Overview

The C-Language Function Call command (CFUNC) calls the user C-language task.



Note: The following versions of system software and programming tool MPE720 are required to use CFUNC commands.

MP2000 Series Machine Controller	Required Version
All models	Ver.2.60 or later

MPE720	Required Version
MPE720 Ver.5	MPE720 Ver.5.38 or later
MPE720 Ver.6	MPE720 Ver.6.04 or later MPE720 Ver.6.04Lite or later

INFO

The Machine Controller MP2000 series embedded C-language package is required to use user C-language tasks.
For details, refer to *Machine Controller MP2000 Series Embedded C-Language Programming Package Development Guide* (manual no.: SIEP C880700 25).

(2) Format

```
CFUNC EXECUTE  OPTION1  OPTION2,  C_NAME  C_ARG1  C_ARG2,
                                          COMPLETE  ERROR  C_RETURN;
```

I/O Definition	No.	Name	I/O Designation	Description
Input	1	EXECUTE	B-VAL	Designates execution of CFUNC function
	2	OPTION1	I-REG	Option designation 1 (For future use)
	3	OPTION2	I-REG	Option designation 2 (For future use)
	4	C_NAME	Address input	Designates the leading register No. (MW/DW address) of the registers where the user c-language function name is stored.
	5	C_ARG.1	Address input	Designates the leading register No. (MW/DW address) to be set for the user C-language function 1st argument.
	6	C_ARG.2	Address input	Designates the leading register No. (MW/DW address) to be set for the user C-language function 2nd argument.
Output	1	COMPLETE	B-VAL	Completes execution of the CFUNC function
	2	ERROR	B-VAL	Error occurrence • C_NAME, C_ARG.1, C_ARG.2 register over the upper/lower limit (The sizes of C_ARG.1 and C_ARG.2 are not taken into consideration.) Note: Detected also when the EXECUTE input is OFF. • The function designated in C_NAME does not exist.
	3	C_RETURN	L-REG	Stores the return value of user C-language function as it is.

(3) Programming Examples

The CFUNC command programming example is shown below.

◀ EXAMPLE ▶

```
ASCII 'cfunc1' DW00010;      "User C-language function name
DW00013 = 0000H;             "NULL code
DB000000 = 1;                "Execute the user C-language function
CFUNC DB000000 0 0,DA00010 DA00002 DA00004, DB000003 DB000004
DL00016;
```

9

Engineering Tool MPE720

This chapter describes the engineering tool MP720, for creating and editing motion programs and sequence programs.

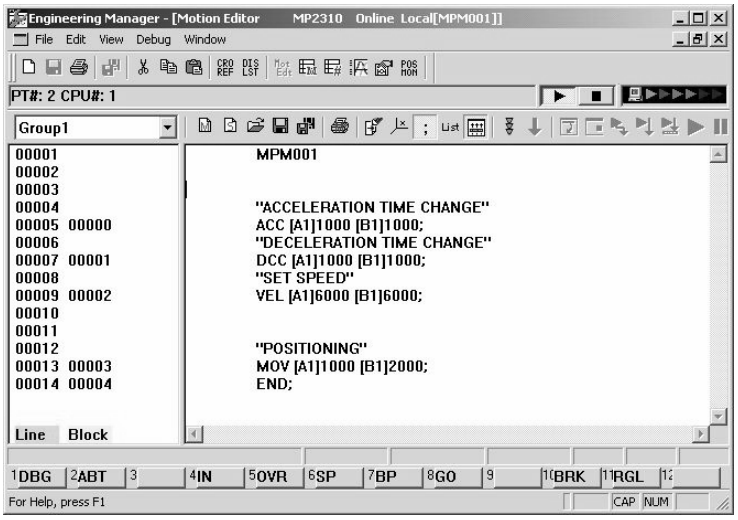
9.1 Motion Editor ----------9-2
9.1.1 Overview ---------- 9-2
9.1.2 Names and Descriptions of Motion Editor Window Components ---------- 9-4
9.2 Command Input Assistant Function ----------9-6
9.2.1 Overview ---------- 9-6
9.2.2 Motion Command Assist Dialog Box Details ---------- 9-8
9.3 Program Execution Registration Function ----------9-12
9.3.1 Overview ---------- 9-12
9.3.2 Program Execution Registry Screen Dialog Box Details ---------- 9-13
9.4 Debug Function ----------9-15
9.4.1 Overview ---------- 9-15
9.4.2 Motion Editor Window during Debugging ---------- 9-16
9.5 Motion Task Manager ----------9-22
9.5.1 Overview ---------- 9-22
9.5.2 Motion Task Manager Window Details ---------- 9-23
9.6 Drive Control Panel ----------9-24
9.6.1 Overview ---------- 9-24
9.6.2 Drive Control Panel Details ---------- 9-26
9.7 Test Run Function ----------9-28
9.7.1 Overview ---------- 9-28
9.7.2 Test Run Window Details ---------- 9-29
9.8 Axis Status and Alarm Monitor ----------9-31
9.8.1 Overview ---------- 9-31
9.8.2 Monitor Window Details ---------- 9-33

9.1 Motion Editor

This section describes the Motion Editor.

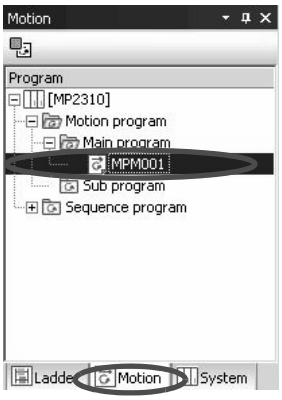
9.1.1 Overview

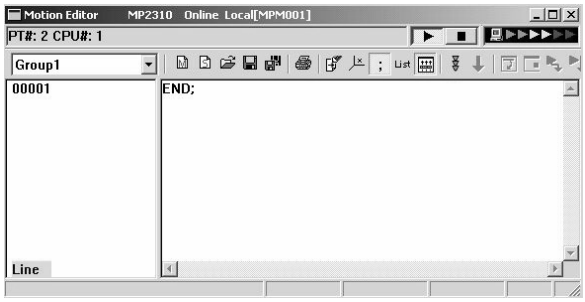
The Motion Editor is a programming tool that is required to create and edit motion programs and sequence programs.
It has the full range of functions required to create and edit these programs, including text editing, compiling (saving), debugging, and monitoring.



There are two ways to start the Motion Editor:

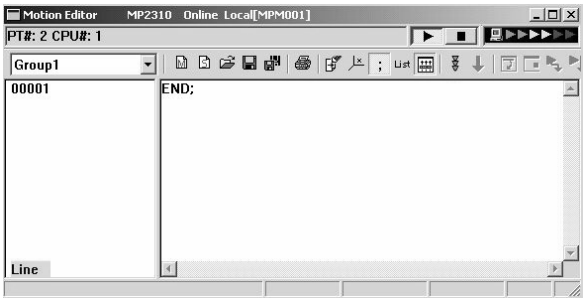
- Double-click the program in the **Motion** subwindow.



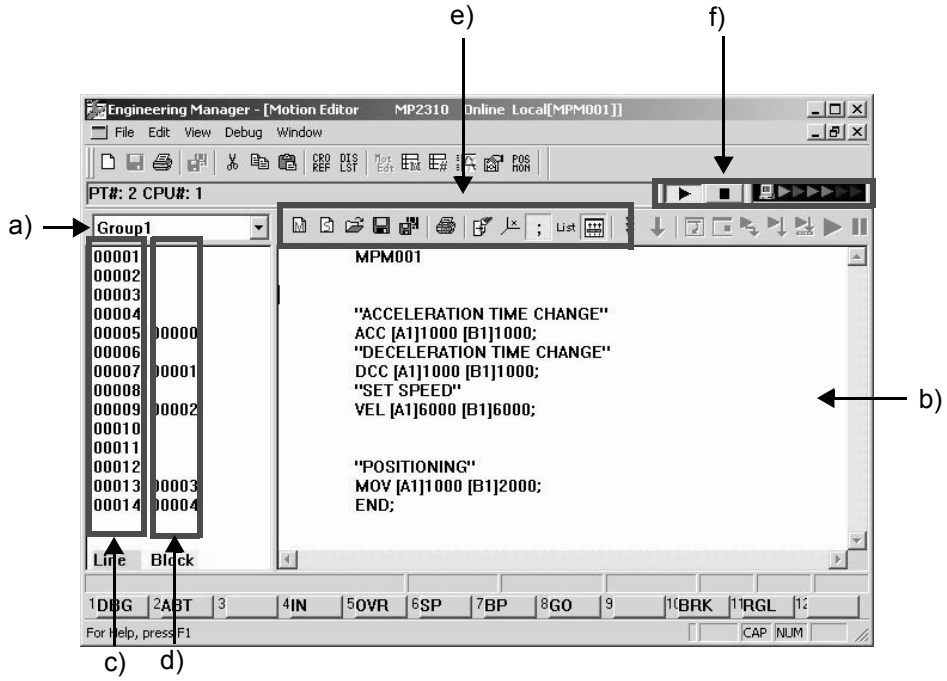


- Click the Mot Edt icon on the toolbar of the **Engineering Manager** window.



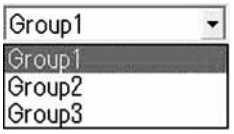


9.1.2 Names and Descriptions of Motion Editor Window Components



a) Group selection (Only for motion programs)

The group names set in the **Group Definition** dialog box are displayed in the drop-down menu. Select a group for which the editing program is to be used.



b) Program editing window

A text editor for programming.

c) Line

Displays program text lines.

d) Block

Displays program blocks.
When an alarm is generated in the motion program, the block where the alarm is generated is reported.

e) Tool icons

The following table shows the icons used to edit programs.

Function	Icon	Key Operation	Description
Cut		Ctrl + X	Deletes the selected range and stores the deleted range in the Clipboard.
Copy		Ctrl + C	Reproduces the selected range in the Clipboard.
Paste		Ctrl + V	Pastes the materials stored in the Clipboard.
Position Monitor		–	Displays the **Position Monitor** dialog box.
New File (Motion programs)		–	Opens a new motion program file.
New File (Sequence programs)		–	Opens a new sequence program file.
Open		–	Displays the **File List** dialog box.
Save		Ctrl + S	Saves the editing program to the hard disk of the computer. During online editing, the editing program will be not only be saved to the hard disk but also downloaded (transferred) to the Machine Controller.
Save & Save to Flash		–	Consecutively executes the following operations. • Saves the editing program to the hard disk of the computer. • Downloads the editing program to the Machine Controller. • Saves the downloaded program to the flash memory. All programs downloaded to the Machine Controller will be saved to the flash memory.
Print		Ctrl + P	Prints out the editing program.
Motion Command Assist		F12	Displays the **Motion Command Assist** dialog box.
Position Teach		–	Displays the **Position Teach** dialog box. The work coordinate system current position of the axis designated in the **Position Teach** dialog box will be inserted on the Motion Editor.
Add/Not Add a Semicolon		–	Used to select whether or not to allow use of the ENT key to add a semicolon (;).
Open Error List Dialog Box		–	Displays the list of save errors (compile errors).
Automatic Scroll		–	Used to select whether or not to scroll the screen in line with the executing line of the program during program execution.

f) Monitor tab and status bar

Used to monitor programs.

Function	Button/ Status Bar	Key Operation	Description
Monitoring Tab		–	Sets whether or not to monitor programs. When the moving button is selected (), the executing program line is displayed in real time.
Monitoring Status Bar		–	Indicates the program monitoring status.

9.2 Command Input Assistant Function

This section describes the motion command input assistant function.

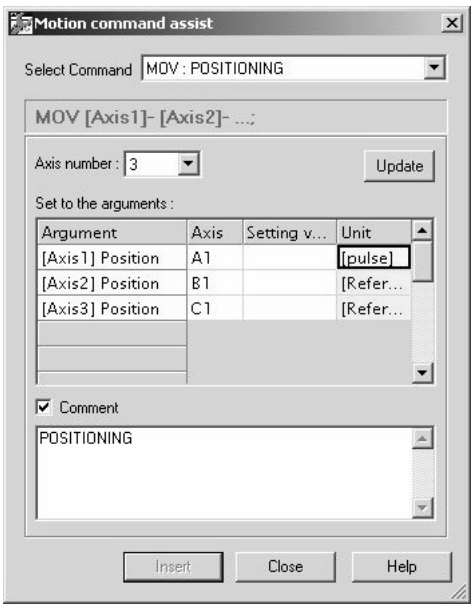
9.2.1 Overview

The command input assistant function helps a programmer to create a required motion program.
The motion commands are written in a textual language called motion language, and each command needs to be written in the specified format. The **Motion command assist** dialog box is provided to make it easy to correctly select and code the commands.
The following versions of MPE720 support the command input assistant function.

MPE720	Supported Version
MPE720 Ver.5	Not supported
MPE720 Ver.6	Ver.6.04 or later Ver.6.04 Lite or later

Note: The command input assistant function can be used for all models of MP2000-series Machine Controllers.

The **Motion command assist** dialog box can be opened from the **Motion Editor** window. There are two ways to open the **Motion command assist** dialog box, as described on the next page.



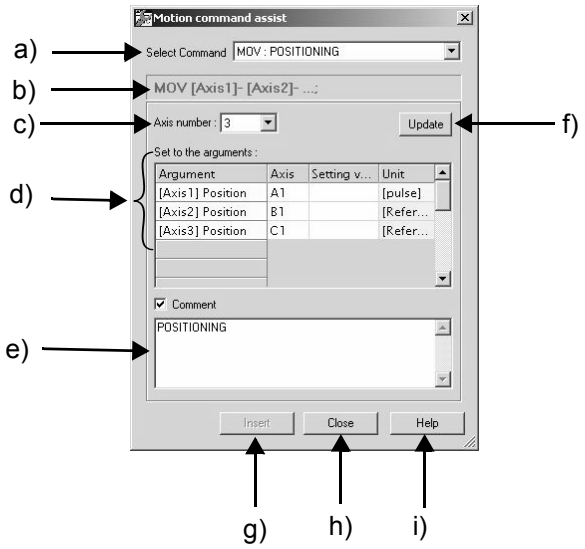
- Select **Motion command assist** from the pop-up menu when you right-click the **Motion Editor** window, or press the function key F12.



- From the right-click menu, select **Insert motion command**, then select the command you want to insert.

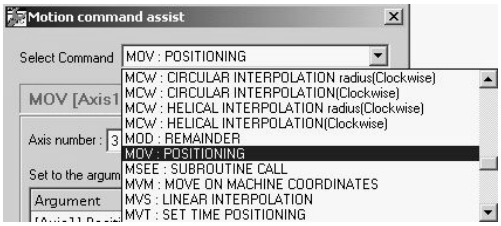


9.2.2 Motion Command Assist Dialog Box Details



a) Select command (Command options)

Click the arrow to open the drop-down menu that contains the commands that can be inserted.



b) Command format

Displays the programming format of the selected command.

◀ EXAMPLE ▶

MOV: Positioning

```
MOV [Axis1]- [Axis2]- ...;
```

+: Add

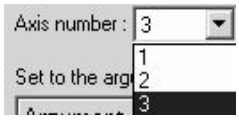
```
ML00106=ML00102 + ML00104;
```

c) Axis number (Number of axes to be controlled)

For axis move commands, the number of axes to be controlled can be selected in the range from 1 to the number of axes set in the **Group Definition** dialog box.
When the number of axes to be controlled is fixed, the fixed number of axes is displayed in the shaded box (unavailable option).

◀ EXAMPLE ▶

MOV: Positioning ••• Select the number of axes to be controlled.



EXM: External positioning ••• The number of axes to be controller is fixed.

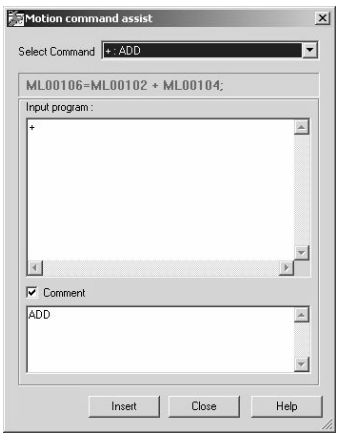


d) Set to the arguments

Set the parameters for the selected command. The setting items are listed below.

Item	Description
Argument	Displays the parameter names to be set as the arguments. The displayed parameter names cannot be edited. For the arguments that can be omitted, **[Can be omitted]** is displayed.
Axis	Displays the logical axis names. Change the logical axes as required.
Setting value	Enter a constant or register as the set value.
Unit	Displays the parameter setting unit. The setting units cannot be edited.

The logical axis names displayed in the **Axis** column are defined in the **Group Definition** dialog box.
The setting units in the **Unit** column are displayed according to the motion parameter settings of each axis. If a setting unit has not been specified, the corresponding **Unit** cell is displayed in yellow. Place the mouse pointer on the **Unit** cell and click the mouse. The help balloon will pop up. Follow the help balloon messages to set the motion parameters.
When the selected command requires neither the settings of number of axes to be controlled nor parameters, the program input field will appear as shown below. Enter the command block referring to the command format displayed above.



e) Comment check box and Comment input field

Selecting the **Comment** check box allows you to insert a comment above the command line. When the **Comment** check box is cleared, the comment input field will be shaded and comments cannot be inserted.

INFO The comment inserting position cannot be changed.

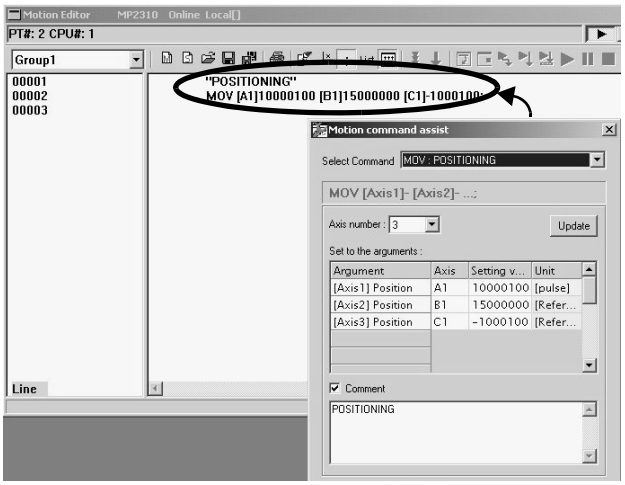
f) Update button

Click the **Update** button to refresh the display in the **Motion command assist** dialog box.

INFO After changing the setting of a unit related motion parameter, click the **Update** button to refresh the display.

g) Insert button

Click the **Insert** button to insert the command edited in the **Motion command assist** dialog box at the pointer position in the **Motion Editor** window.

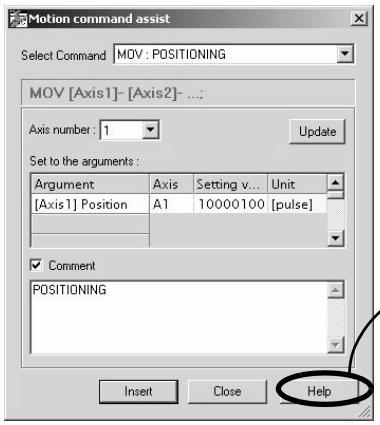


h) Close button

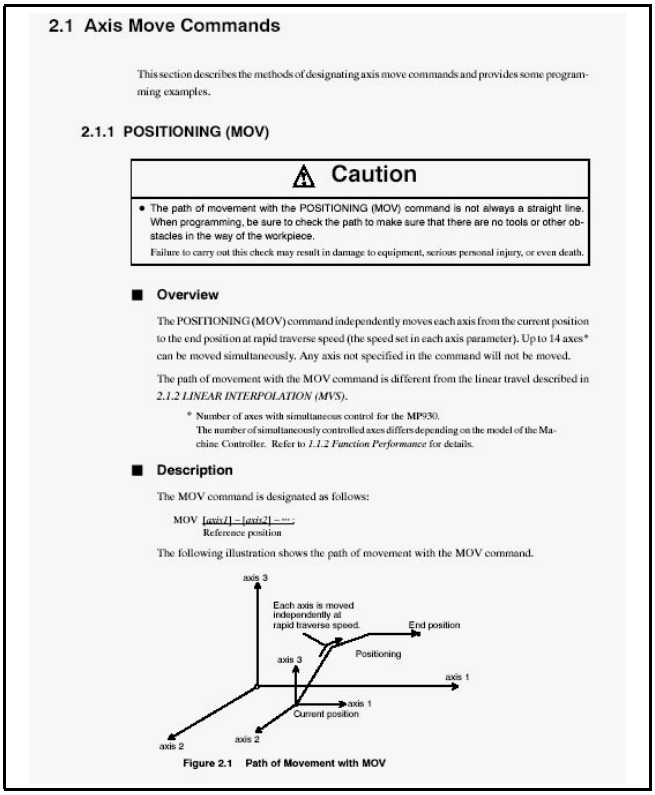
Click the **Close** button to close the **Motion command assist** dialog box.

i) Help button

Click the **Help** button to display a description of the selected command.



Jumps to the help page of the selected command



2.1 Axis Move Commands

This section describes the methods of designating axis move commands and provides some programming examples.

2.1.1 POSITIONING (MOV)

Caution

- The path of movement with the POSITIONING (MOV) command is not always a straight line. When programming, be sure to check the path to make sure that there are no tools or other obstacles in the way of the workpiece.

Failure to carry out this check may result in damage to equipment, serious personal injury, or even death.

■ **Overview**

The POSITIONING (MOV) command independently moves each axis from the current position to the end position at rapid traverse speed (the speed set in each axis parameter). Up to 14 axes* can be moved simultaneously. Any axis not specified in the command will not be moved.

The path of movement with the MOV command is different from the linear travel described in *2.1.2 LINEAR INTERPOLATION (MVS)*.

* Number of axes with simultaneous control for the MP930.
The number of simultaneously controlled axes differs depending on the model of the Machine Controller. Refer to *1.1.2 Function Performance* for details.

■ **Description**

The MOV command is designated as follows:

MOV [axis1] – [axis2] – ···;
Reference position

The following illustration shows the path of movement with the MOV command.

Figure 2.1 Path of Movement with MOV

9.3 Program Execution Registration Function

This section describes the program execution registration function of M-EXECUTOR.

9.3.1 Overview

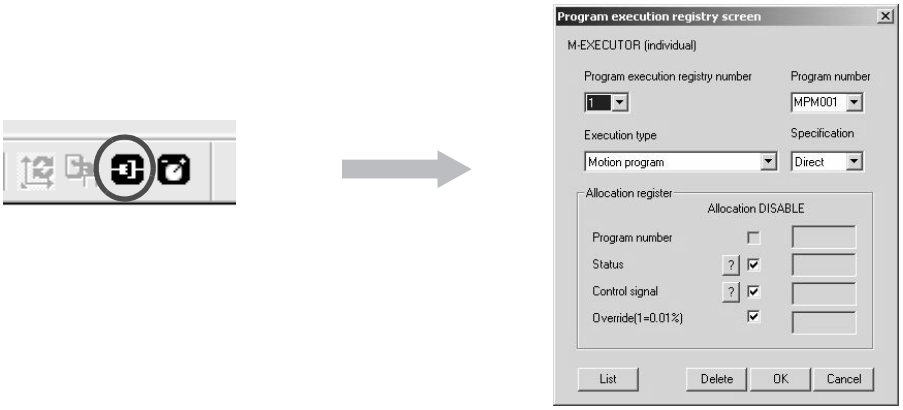
The created motion programs and sequence programs need to be registered in the MP2000 system to run. The **Program execution registry screen** dialog box is provided to make it easy to register the created motion programs and sequence programs in the MP2000 system.

The following versions of MPE720 support the program execution registration function.

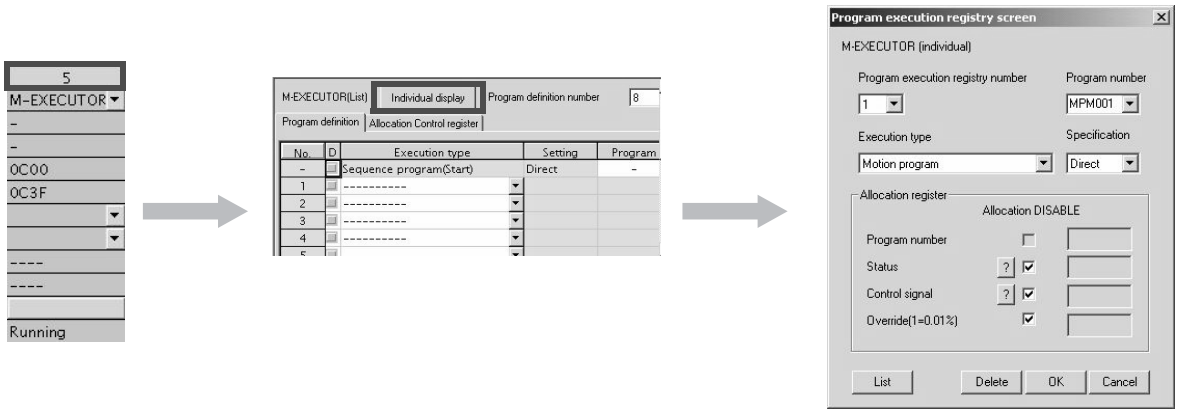
MPE720	Supported Version
MPE720 Ver.5	Ver5.38 or later
MPE720 Ver.6	Ver6.04 or later Ver6.04 Lite or later

There are two ways to open the **Program execution registry screen** dialog box:

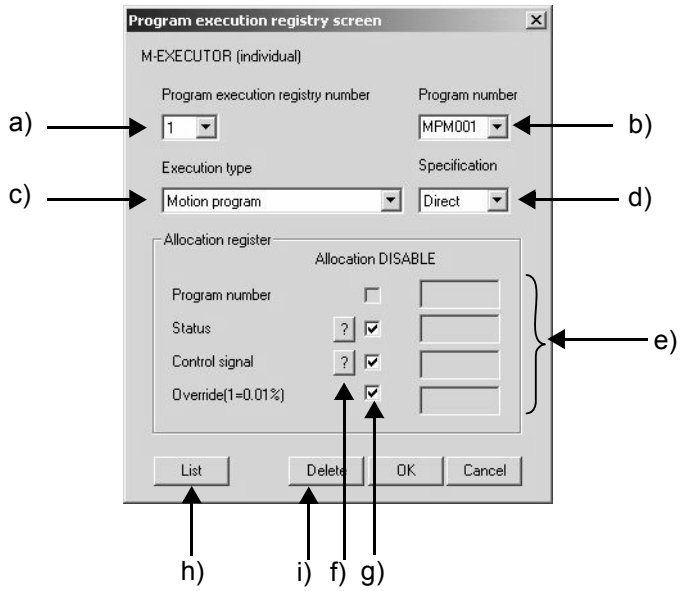
- Click the icon in the **Motion Editor** window.



- Open the M-EXECUTOR details window from the **Module Configuration Definition** window, and then click the Individual display icon.



9.3.2 Program Execution Registry Screen Dialog Box Details



a) Program execution registry number

Select the program execution registration number.
Programs will be executed in order, from the smallest program execution registration number.

b) Program number

Set the program number.

c) Execution type

Select a program execution type from the drop-down menu:

Execution Type	Program to be Executed	Execution Conditions
Sequence program (Start)	Sequence programs	At power ON (Executed once at power ON.)
Sequence program (L scan)		Cyclic start (Executed in low-speed scan cycle.)
Sequence program (H scan)		Cyclic start (Executed in high-speed scan cycle.)
Motion program	Motion programs	When the Program operation start request bit of control signal turns ON. (Executed when the Program operation start request bit turns ON.)

d) Specification

Select the program designation method: Direct or Indirect
The applicable designation methods differ depending on whether the program is a motion program or a sequence program.

Designation Method	Motion Programs	Sequence Programs	Description
Direct	Applicable	Applicable	Designate the program number. Example: MPM001, SPM002
Indirect	Applicable	N/A	Designate the register number that stores the program number. Example: OW0C0C (When 1 is stored in OW0C0C, MPM001 is executed.)

e) Register allocation

Allocate the registers. The allocated registers exchange data in real time with the M-EXECUTOR control registers. I, O, and M registers can be allocated.

f) Status, Control signal (? icons)

Click the corresponding icon to display the bit assignment of the status flag or control signal.





g) Allocation DISABLE

Click to select or clear the corresponding check box to enable or disable the allocated register. When the check box is cleared, the allocated register is enabled.

h) List button

Click the **List** button to display the **M-EXECUTOR (list)** window.

i) Delete button

Delete the definition.

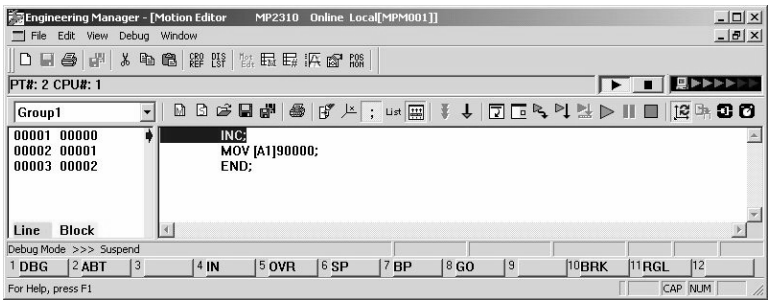
9.4 Debug Function

This section describes the debug function.

9.4.1 Overview

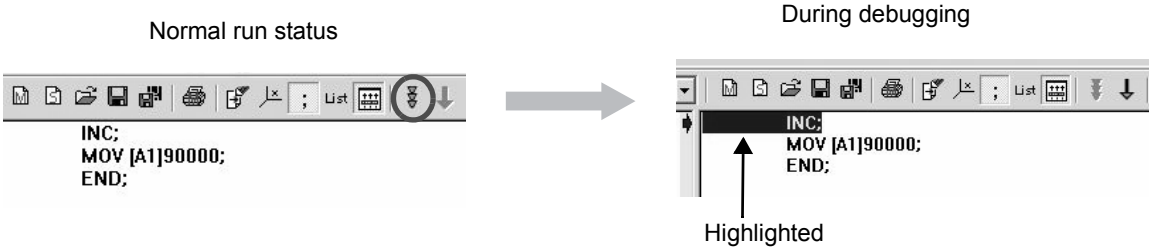
The debug function debugs motion programs and sequence programs.
Various functions including program pause, break point setting, and step-by-step execution (single block execution) are provided to check the created program operation.

During debugging, the executing program line is highlighted on the screen as shown below.



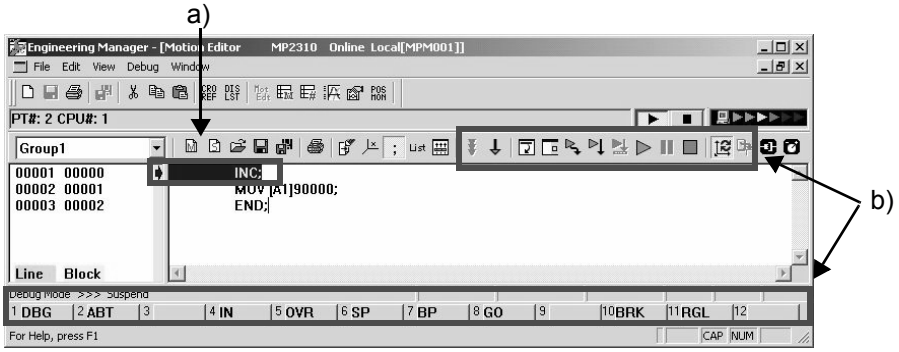
Note: The debug function can be used with all MP2000-series Machine Controller models.

To start debugging, connect the MPE720 online with the Machine Controller, and click the icon on the **Motion Editor** window.
During debugging, the executing program line is highlighted on the screen as shown below.



INFO Before debugging, the programs must be registered in the **Program execution registry screen** dialog box.

9.4.2 Motion Editor Window during Debugging



a) Executing program line

The executing program line is highlighted in blue.
If a motion program alarm is generated in the executing program line, the executing program line is highlighted in red. Refer to *10.2.4 Motion Program Alarm Codes* for information on motion program alarms.

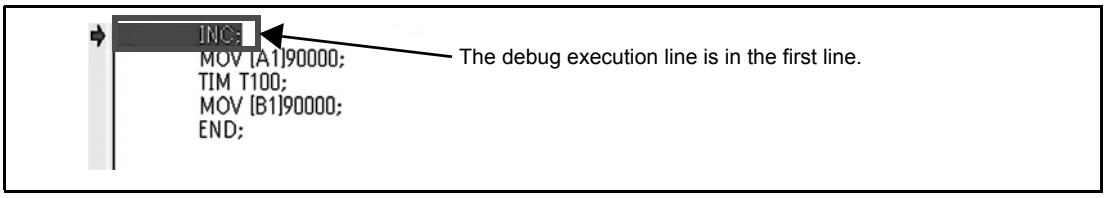
b) Tool icons and function keys

The following table shows the icons, menu commands, and function keys to be used for a debug operation.

Function	Icon	Key Operation	Description	Motion Programs	Sequence Programs
Debug mode		F1	Starts the program in debug mode.	Available	Available
Normal run mode		F11	Cancels the debug mode. The program continuously runs in normal run mode.	Available	Available
Shift execution start point		F6	Shifts the execution start line (start point).	Available	Available
Set/Delete break point		F7	Sets or deletes a break point. A break point is described in the program.	Available	Available
Step in		F4	Executes one command block. For an MSEE or SSEE command, execution will jump to the start line of the specified subprogram.	Available	Available
Step over		F5	Executes one command block. For an MSEE or SSEE command, executes the specified subprogram and then executes the next block of the MSEE or SSEE command.	Available	Available
Execute		F8	Continuously executes the motion program lines in debug mode until the break point.	Available	Available
Break		F10	Pauses motion program execution in debug mode.	Available	Available
Forced end		F2	Forcibly stops motion program execution.	Available	N/A
Update current position		–	Updates the current position coordinates.	Available	N/A
Setting the execute task MSEE Call Stack		–	Sets the parallel number, nesting level, and task of the selected subprogram.	Available	Available
Set/Delete Breakpoint	–	–	Enables or disables the break point. This command can be selected from the **Debug** menu or the drop-down menu that will appear when right clicking.	Available	Available
Add Register	–	–	Adds the selected register to the Watch Page of Quick Reference. This command can be selected from the drop-down menu that will appear when right clicking.	Available	Available

- Debug mode

Click this icon to switch the operation mode to the debug mode and start debugging from the first line of the program.



INFO

The debug start line when the operation mode is switched to the debug mode differs, depending on whether you are using the motion program or the sequence program:

■ Motion Programs

- When debug mode is selected for a motion program that is not running
  Debugging starts from the first line of the program, as shown in the above example.
- When debug mode is selected for a running motion program
  When the mode is switched to the debug mode while an axis is moving, debugging will start from the next command block after the axis movement completes.

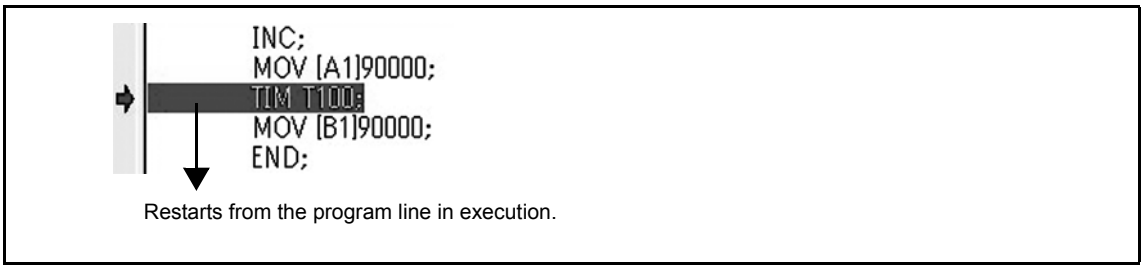
■ Sequence Programs

- When debug mode is selected for a sequence program that is not running.
  Debugging is not possible in this case.
- When debug mode is selected for a running sequence program.
  Debugging starts from the first line of the program, as shown in the above example.

- Normal run mode

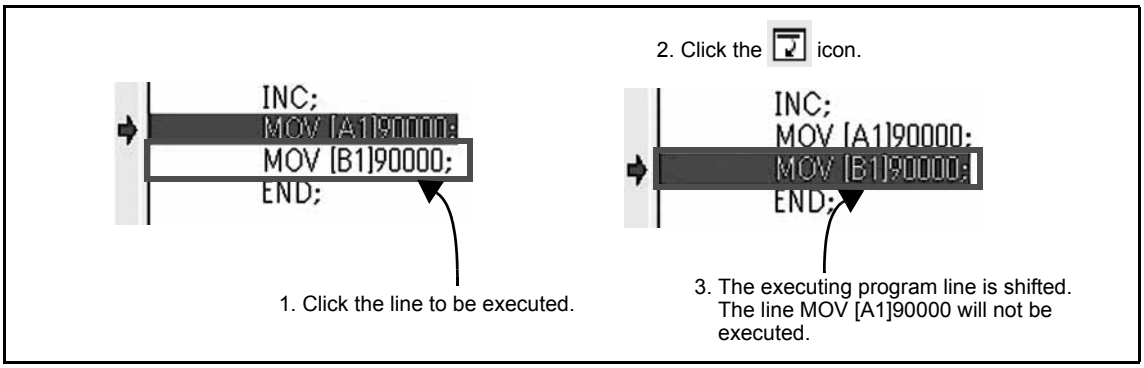
Click this icon to switch the operation mode from debug mode to normal run mode. The debugging will be cancelled and program execution will restart from the currently executing line.

All the break points that have been set will be deleted.



- Shift start point

Shift the executing program line to the selected line.



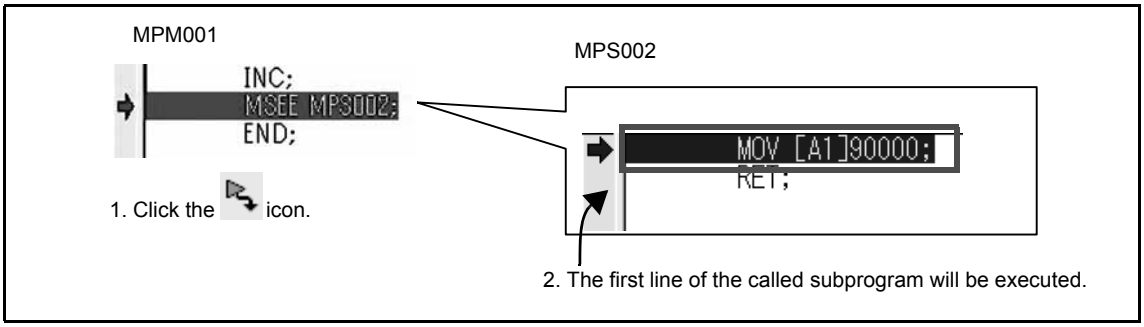
- Set/Delete break point

Click this icon to set a break point. A maximum of four break points can be set.
Clicking the button of the line for which a break point has been already set will delete the break point.



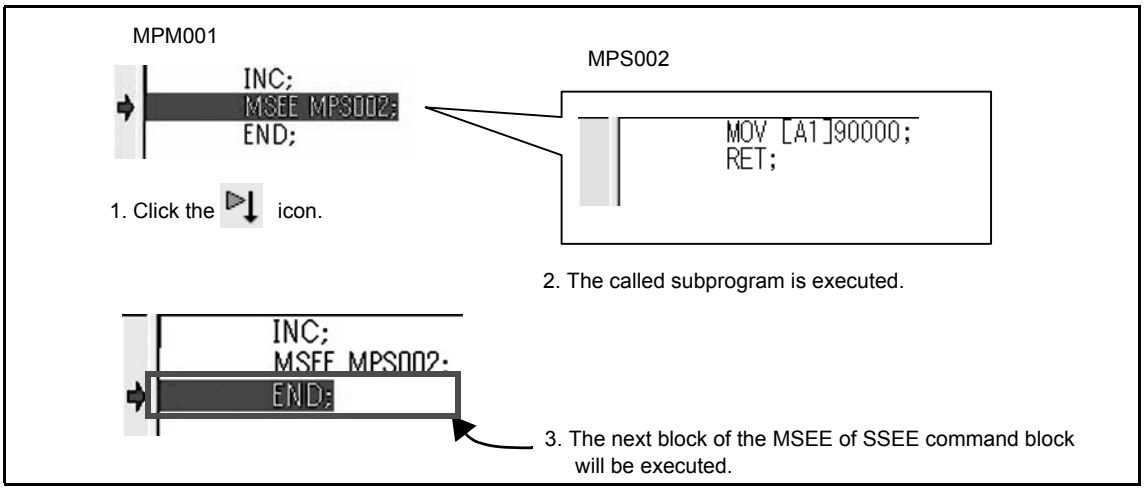
- Step in

Click this icon to execute one program line.
When this icon is clicked while an MSEE or SSEE command block is being executed, execution will jump to the first line of the called subprogram.



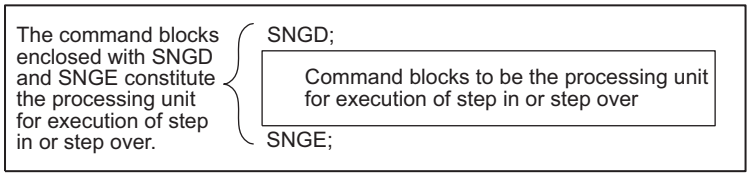
- Step over

Click this icon to execute one program line.
When this icon is clicked while an MSEE or SSEE command block is being executed, the called subprogram will be executed and then execution will proceed to the next block of the MSEE or SSEE command block.



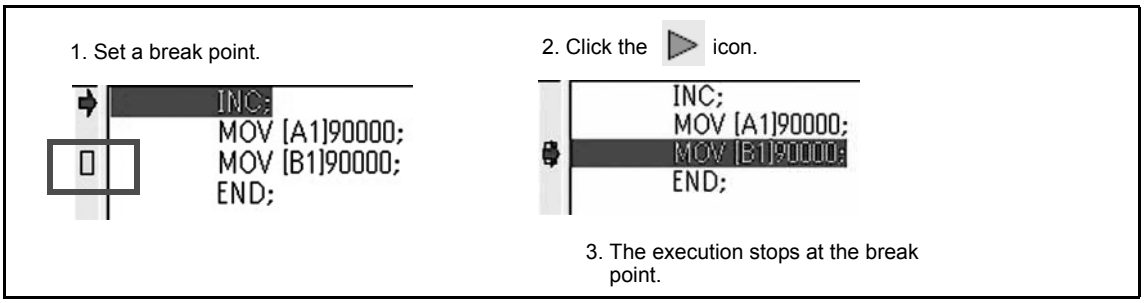
INFO

More than one process can be specified as the processing unit for execution of the Step in and Step over by using SNGD and SNGE commands.



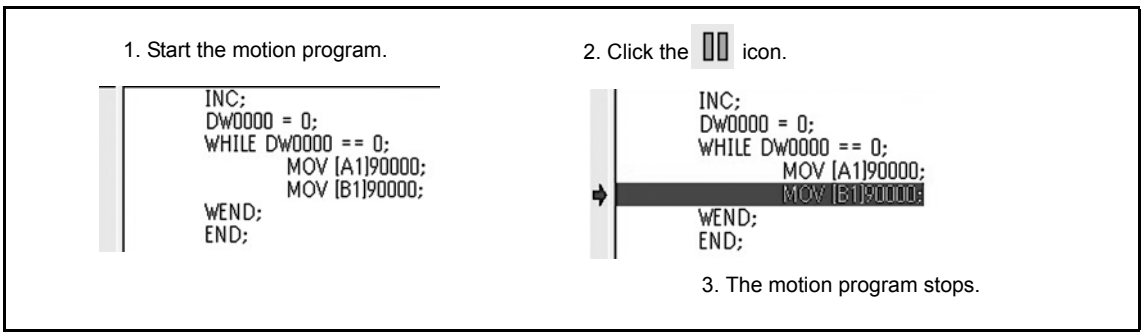
- Execute

Click this icon to continuously execute program lines. When the break point is reached, the execution will stop.



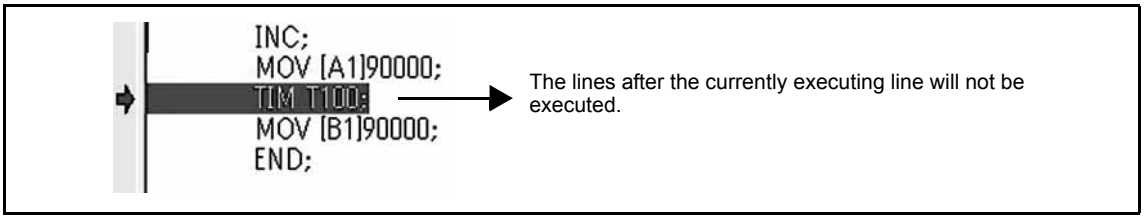
- Break

Click this icon to pause a program in debug mode. To restart the program, click the **Execute** icon.



- Forced end

Click this icon to forcibly end the execution of a program in debug mode.

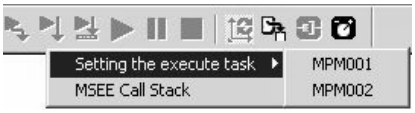


- Update current position

This icon has the same function as the PLD command. While this icon is selected, the operation of the PLD command is processed by the system when using the **Step in**, **Step over**, and **Execute** icons.
Refer to *8.3.3 Program Current Position Update (PLD)* for details on PLD command.

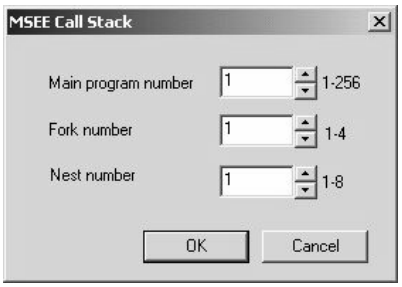
- Setting the execute task (Only for subprograms)

Set the subprogram information used for monitoring or debugging subprograms. When the **Setting the execute task** command is selected, the currently running main programs will be displayed. Select the main program that calls the subprogram.



- MSEE call stack (Only for subprograms)

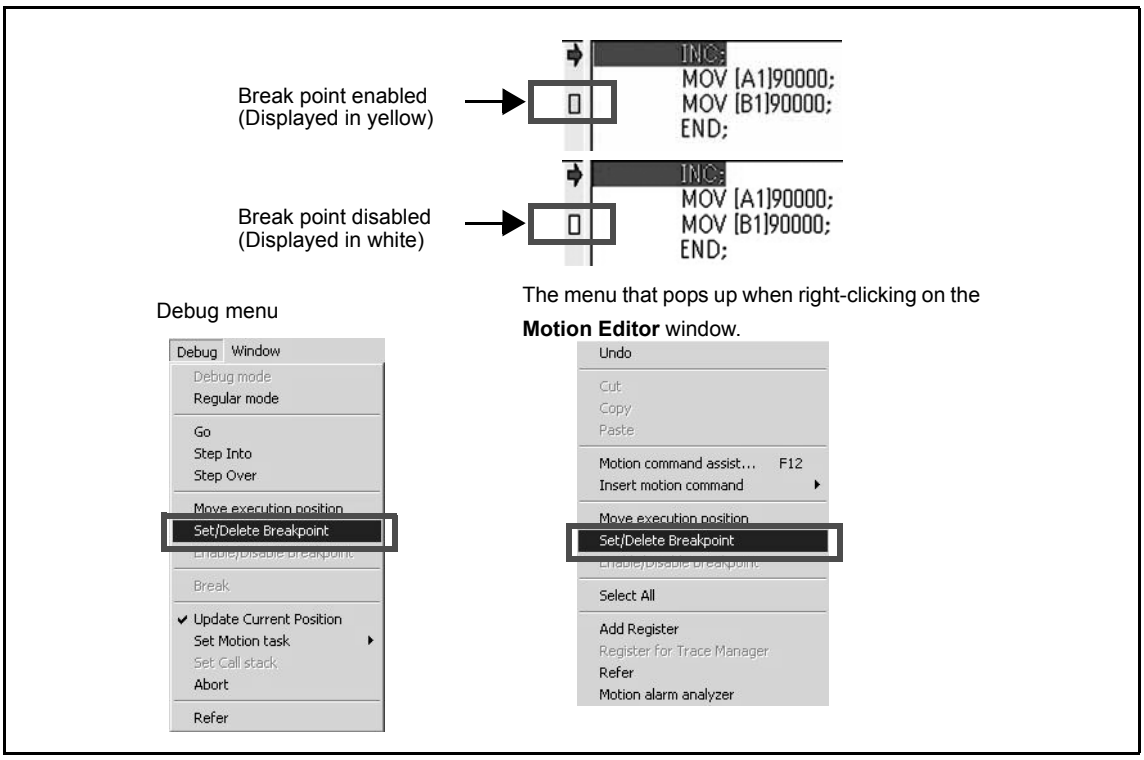
Set more detailed subprogram information than the **Setting the execute task** command.



Item	Description
Main program number	Set the main program number that calls the subprogram.
Fork number	Set the parallel number of the main program that calls the subprogram. Example: Set 3 when monitoring and debugging subprogram MPS004. MPM001 PFORK Label1 Label2 Label3 Label4; Label1: " Parallel 1 MSEE MPS002; JOINTO LabelX; Label2: " Parallel 2 MSEE MPS003; JOINTO LabelX; Label3: " Parallel 3 MSEE MPS004; JOINTO LabelX; Label4: " Parallel 4 MSEE MPS005; JOINTO LabelX; LabelX: PJOINT; .... END;
Nest number	Set the nesting level of the subprogram call. Example: Set 2 when monitoring and debugging subprogram MPS003. MPM001 MW0000=1; MSEE MPS002; .... END; MPS002 (Nest 1) MW0000=2; MSEE MPS003; .... RET; MPS003 (Nest 2) MW0000=3; .... RET;

- Set/Delete Breakpoint

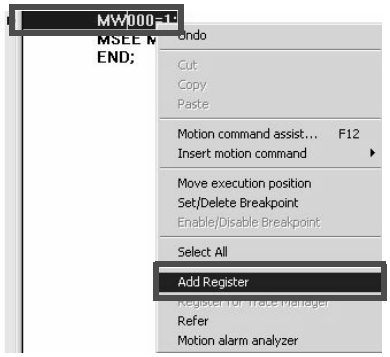
Enables or disables the break point.



- Add Register

The register displayed on the Motion Editor can be registered to the Watch Page of Quick Reference. The values of the registered registers can be monitored on the Watch Page.

1. Right-click on the register to be monitored. Select **Add Register** from the pop-up menu.



2. The register will be added to the Watch Page of Quick Reference.

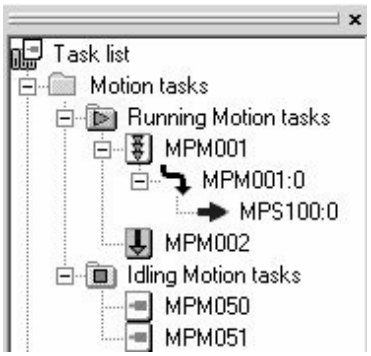


9.5 Motion Task Manager

This section describes the Motion Task Manager.

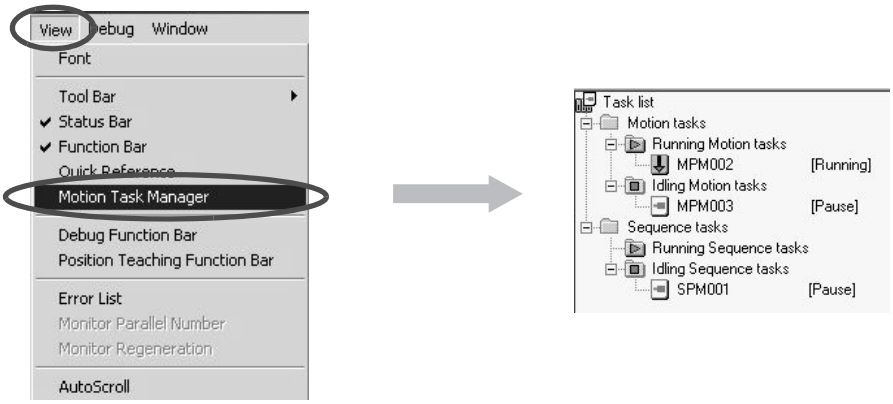
9.5.1 Overview

The Motion Task Manager is used to view the list of motion tasks, and to monitor task execution status. The running tasks and idle tasks are displayed in a tree structure in the **Motion Task Manager** window.

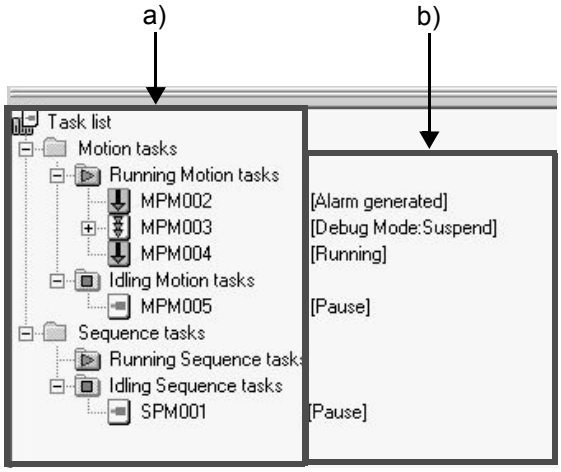


Note: The Motion Task Manager can be used with all MP2000-series Machine Controller models.

To start the Motion Task Manager, select ***View - Motion Task Manager*** in the **Motion Editor** window.



9.5.2 Motion Task Manager Window Details

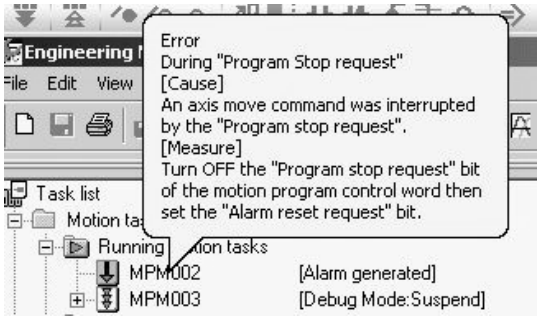


a) Task execution status is displayed in a tree structure

The execution status of each motion program and sequence program is displayed in a tree structure.
Double-click the program to open the corresponding program in the **Motion Editor** window.

b) Program status display

The program status is displayed.

Display	Status
Pause	The program execution is paused.
Debug Mode	The program is being debugged.
Debug Mode: Suspend	The program debugging is suspended at a break point.
Running	The program is running.
Alarm generated	An alarm has occurred in the motion program. Place the mouse pointer on the program name to view the alarm details. 

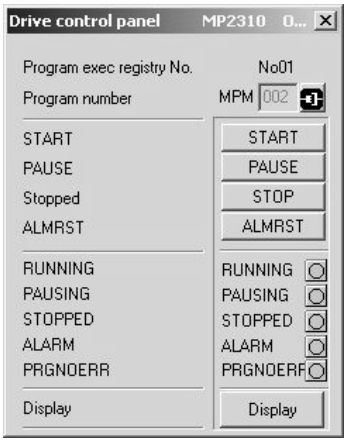
9.6 Drive Control Panel

This section describes the Drive Control Panel.

9.6.1 Overview

To execute the created motion program, the program needs to be registered in the MP2000 system and the program start request must be issued using the user application.
If you want to run the created motion program before creating the user application, use the **Drive control panel** to carry out the program trial run.
Commands such as program start request, stop request, and alarm reset request can be issued from the **Drive control panel**.
The following versions of MPE720 support the **Drive control panel**.

MPE720	Supported Version
MPE720 Ver.5	Ver. 5.38 or later
MPE720 Ver.6	Ver. 6.04 or later Ver. 6.04 Lite or later

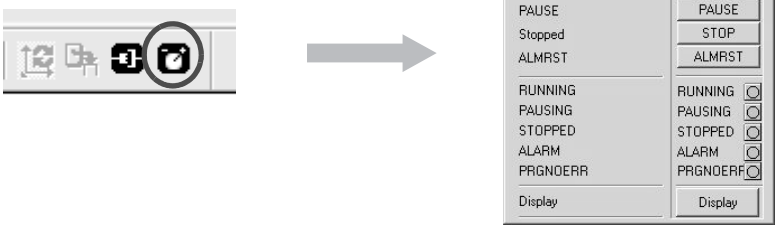


Note: The **Drive control panel** does not have the break point setting or step-by-step execution (single block execution) functions that the debug function has.

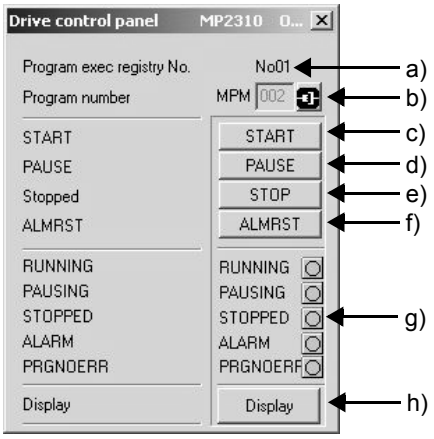
⚠ CAUTION

- Be sure the area is safe before starting a motion program by using the Drive control panel.
  The axes will start moving as the motion program runs.
- Do not overwrite the motion program control register by using a sequence program or ladder program.
  Doing so may disable the control from the Drive control panel.
- Do not simultaneously execute axes move commands for one axis from more than one program.
  Doing so may result in an adverse movement of the axis.

Click the icon in the **Motion Editor** window to start the **Drive control panel**.



9.6.2 Drive Control Panel Details



a) Program exec registry No.

The program execution registry number of the program for trial run is displayed.
The program execution registry number must be set in the **Program execution registry screen** dialog box of the M-EXECUTOR in advance.

b) Program number

The program number for trial run is displayed.
The program number must be set in the **Program execution registry screen** dialog box of M-EXECUTOR in advance.

c) START button

Click the **START** button to start the trial run.

d) PAUSE button

Click the **PAUSE** button to pause the trial run.

e) STOP button

Click the **STOP** button to stop the trial run.

f) ALMRST button

Click the **ALMRST** button to reset the alarm after an alarm has been generated.

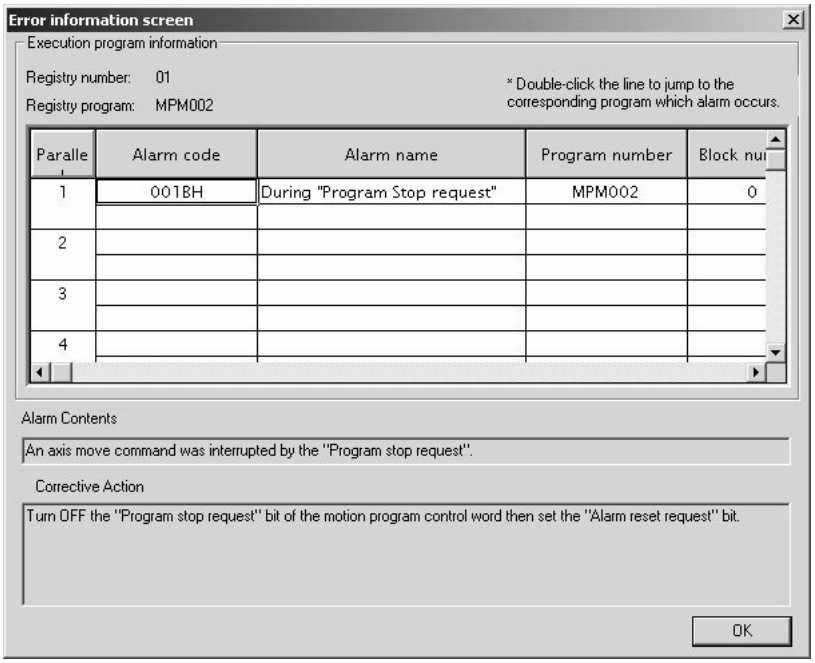
g) Trial run status display

The corresponding status indicator LED lights up to indicate the trial run status.

- RUNNING: Lights up when the **START** button is clicked.
- PAUSING: Lights up when the **PAUSE** button is clicked.
- STOPPED: Lights up when the **STOP** button is clicked.
- ALARM: Lights up when a program alarm occurs.
- PRGNOERR: Lights up when a program number error occurs (the program number is outside the allowable range).

h) Display button

Click the **Display** button to open the **Error information screen** window. Refer to *10.2.4 Motion Program Alarm Codes* for details on the errors and the **Error information screen** window. An example of an error information window is shown below.



9.7 Test Run Function

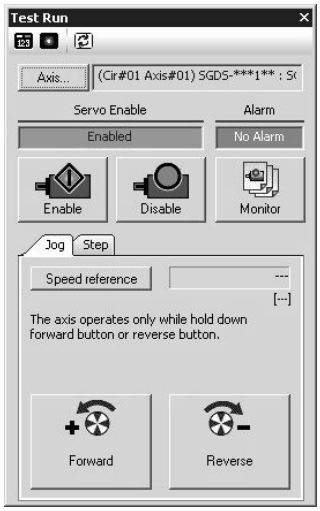
This section describes the Test Run function.

9.7.1 Overview

The Test Run function is used to perform a test run of axes that are connected to the MP2000-series Machine Controller from the **Test Run** window. The servo ON, servo OFF, JOG operation, and STEP commands can be executed without using programs.
The following versions of MPE720 support the Test Run function.

MPE720	Supported Version
MPE720 Ver.5	Not supported
MPE720 Ver.6	Ver6.04 or later Ver6.04 Lite or later

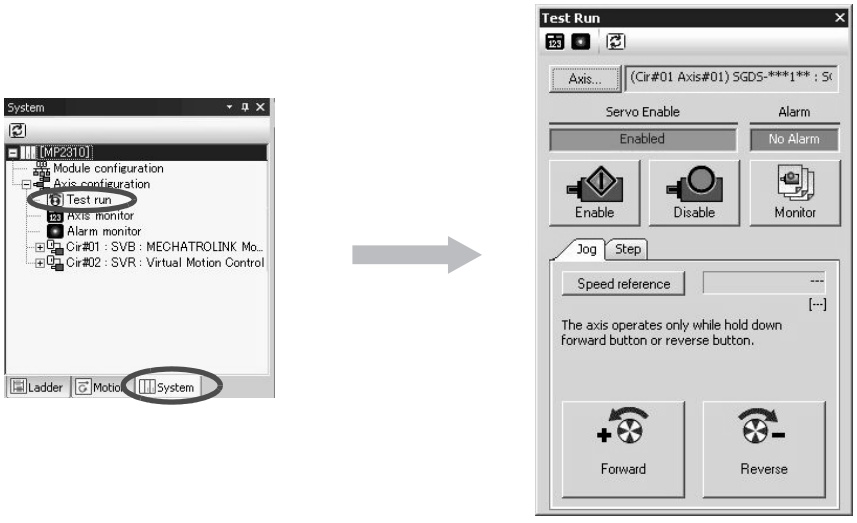
Note: The Test Run function can be used with all MP2000-series Machine Controller models.



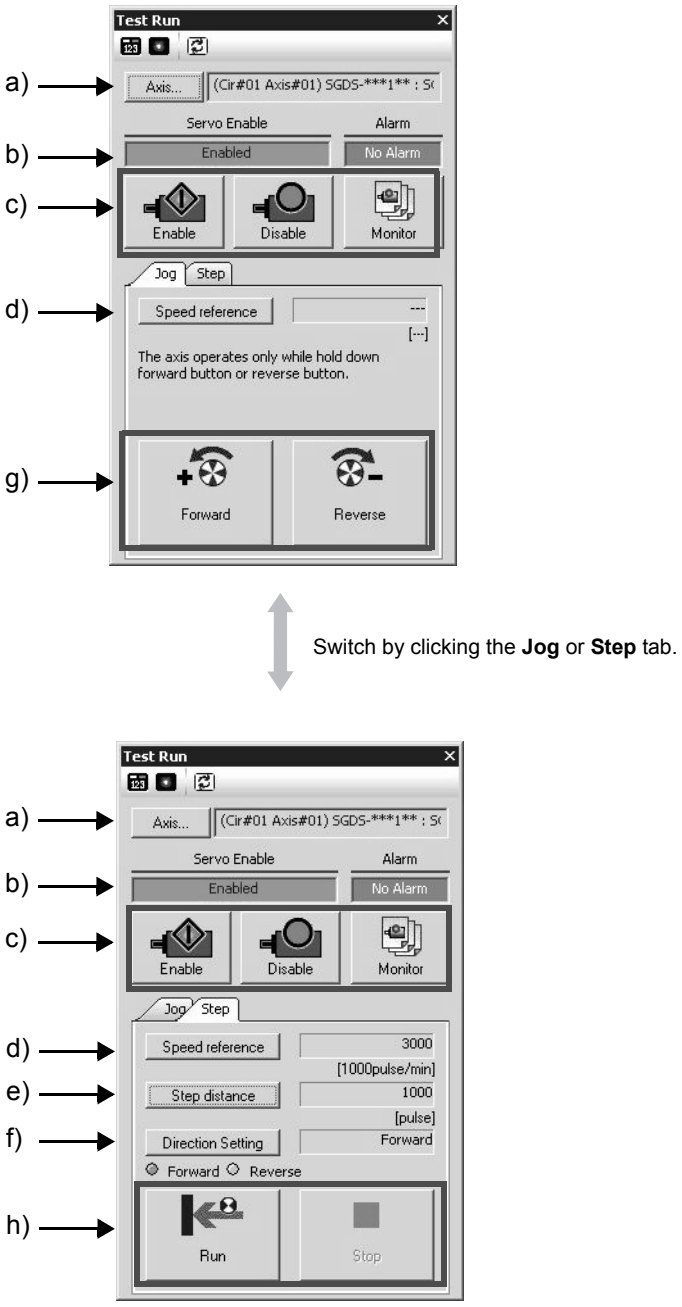
⚠ CAUTION

- Make sure the area is safe before moving the axes by using the Test Run function.
- Before starting operation, be sure to prepare emergency stop measures to stop axis movement whenever necessary.
- Before moving the axes by using the Test Run function, stop all ladder programs and motion programs that are being executed.

Double-click **Test run** in the **System** subwindow to open the **Test Run** window.



9.7.2 Test Run Window Details



a) Axis

Select the axis for the test run.

b) Servo Enable, Alarm

The servo status (**Enabled** [servo ON] or **Disabled**) [servo OFF] and the axis alarm status (**Alarm** or **No Alarm**) are displayed.

c) Enable, Disable, and Monitor buttons

Click the **Enable** button to turn ON the servo, click the **Disable** button to turn it OFF. Special care must be taken for this operation because it will change the motion setting parameter.
Click the **Monitor** button to display the details of an axis alarm.

d) Speed reference

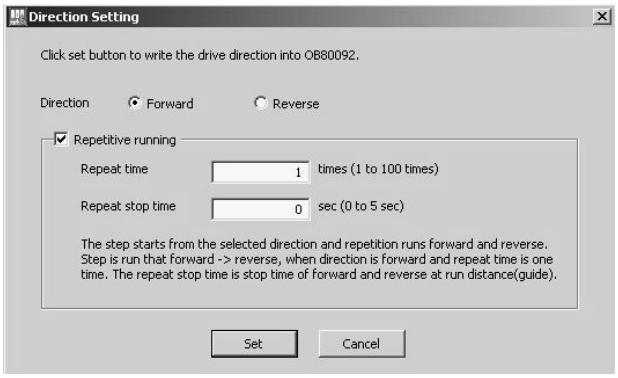
Set the speed reference value. Special care must be taken for this operation because it will change the motion setting parameter.

e) Step distance (In the **Step** tab page)

Set the step movement amount for step operation. Special care must be taken for this operation because it will change the motion setting parameter.

f) Direction Setting (In the **Step** tab page)

Click the **Direction Setting** button. The **Direction Setting** dialog box used to set the axis operation direction for step operation will appear.
Select **Forward** or **Reverse** of the **Direction** to specify the axis operation direction. Special care must be taken for this operation because it will change the motion setting parameter.
A repetitive run for step operation can also be set and specified in this dialog box.



g) Forward and Reverse buttons (In the **Jog** tab page) for JOG operation

Click the **Forward** or **Reverse** button to start JOG operation.
The specified axis keeps moving in the corresponding direction as long as the button is being pressed.
The axis will stop when the button is released.

h) Run and Stop buttons (In the **Step** tab page) for STEP operation

Click the **Run** button to start one step operation of the specified axis. Unlike with the jog operation, the button does not need to be continuously pressed.
When the **Repetitive running** check box is selected in the **Direction Setting** dialog box, the step operation is repeated for the specified number of times, and then the axis will stop. During a repetitive run, the axis can be stopped by clicking the **Stop** button.

9.8 Axis Status and Alarm Monitor

This section describes the axis operation monitor and axis alarm monitor.

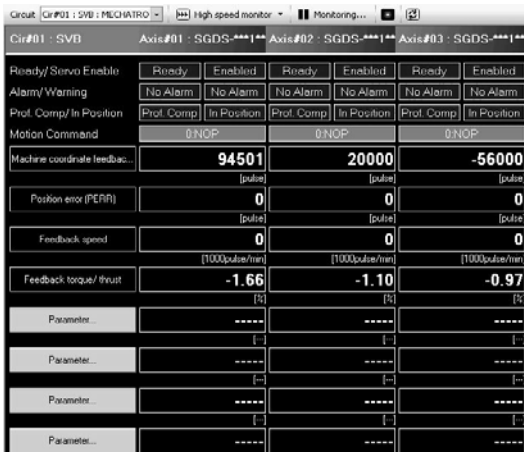
9.8.1 Overview

The **Axis Monitor** displays the operation status of the axes connected to the MP2000-series Machine Controller. The axis status (**Ready/Servo Enable** [Servo ON], **Alarm/Warning**, **Prof. Comp** [Distribution Completed]/**In Position** [Positioning Completed], and **Motion Command**) and the user selected monitoring parameter status are displayed onscreen.
The **Alarm Monitor** displays the alarm information of the axes connected to the MP2000-series Machine Controller.
The following MPE720 versions support the **Axis Monitor** and **Alarm Monitor**.

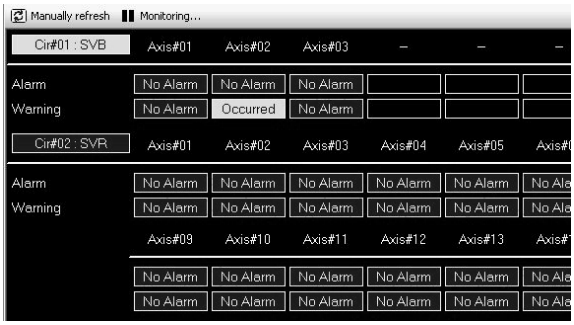
MPE720	Supported Version
MPE720 Ver.5	Not supported
MPE720 Ver.6	Ver. 6.04 or later Ver. 6.04 Lite or later

Note: The Axis Monitor and Alarm Monitor functions can be used with all MP2000-series Machine Controller models.

Axis Monitor



Alarm Monitor

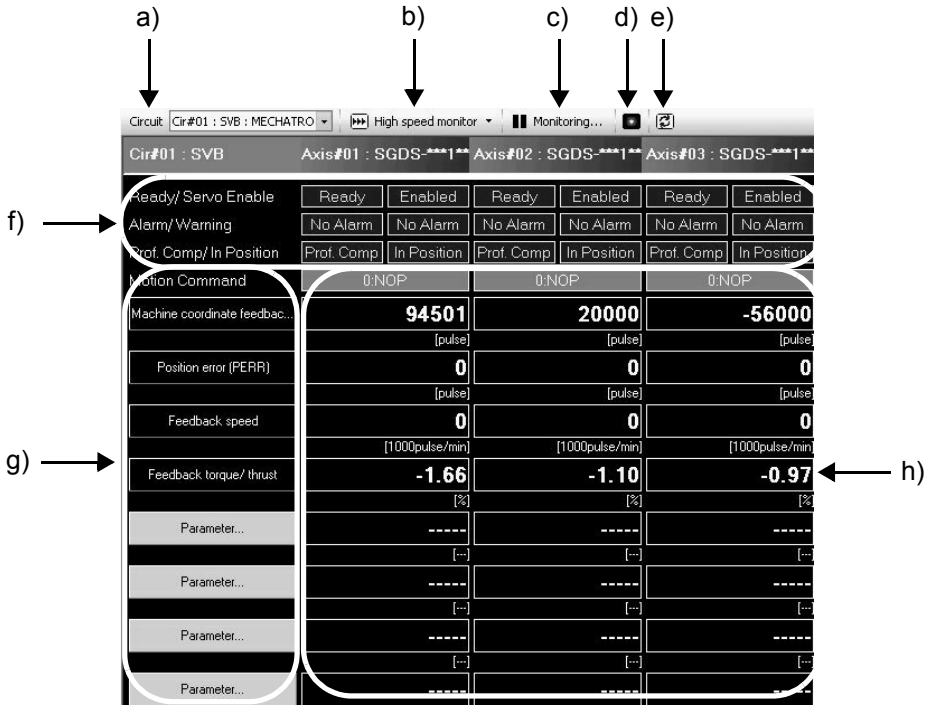


Double-click **Axis monitor** or **Alarm monitor** in the **System** subwindow to start the axis monitor or alarm monitor.



9.8.2 Monitor Window Details

(1) Axis Monitor

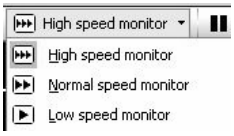


a) Circuit

Select the circuit whose motion monitoring parameters are to be displayed.

b) Monitoring cycle selection

Select the monitoring cycle.



c) Pausing/Starting monitoring

Click this button to start or pause monitoring.

d) Axis alarm monitor

Click this icon to open the **Alarm Monitor** window.

e) Refresh

Click this icon to refresh the display to the latest axis status.

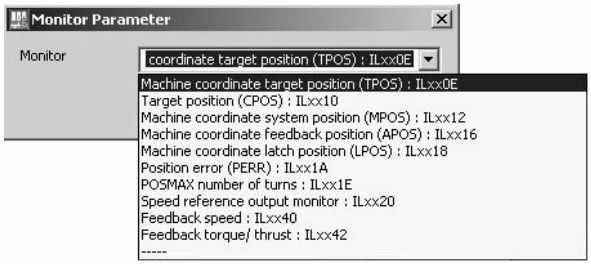
f) Axis status

Ready/Servo Enable (Servo ON), **Alarm/Warning**, **Prof. Comp** (Distribution Completed)**/In Position** (Positioning Completed), **Motion Command** status are displayed. The display will change according to the status.

g) Motion monitoring parameter selection

A maximum of eight motion monitoring parameters can be selected.
Machine coordinate reference position (APOS), Position error (PERR), Feedback speed, and Feedback torque/thrust are displayed as defaults.

Click the Parameter... button, and select a desired monitoring parameter from the **Monitor** drop-down menu in the **Monitor Parameter** dialog box that will appear as shown below.



Monitoring parameter in **Monitor** pull-down list

Monitoring Parameters	Register	Unit
Machine coordinate target position (TPOS)	ILxx0E	Reference unit
Target position (CPOS)	ILxx10	Reference unit
Machine coordinate system position (MPOS)	ILxx12	Reference unit
CPOS for 32bit	ILxx14	Reference unit
Machine coordinate feedback position (APOS)	ILxx16	Reference unit
Machine coordinate latch position (LPOS)	ILxx18	Reference unit
Position error (PERR)	ILxx1A	Reference unit
POSMAX number of turns	ILxx1E	rev
Speed reference output monitor	ILxx20	pulse/s
Feedback speed	ILxx40	Selected speed unit
Feedback torque/thrust	ILxx42	Selected torque unit

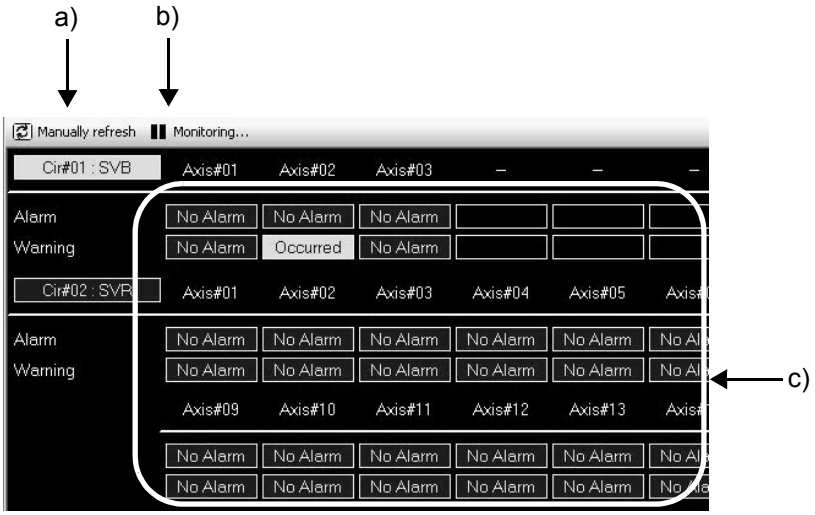
INFO

If you want to set a monitoring parameter not included in the **Monitor** pull-down list, directly enter the register number (IW8000 for example) in the dialog box.

h) Monitoring parameter status

The status of the selected monitoring parameters is displayed.

(2) Alarm Monitor



a) Manually refresh

Click this icon to refresh the alarm and warning information.

b) Pausing/Starting monitoring

Click this button to start or pause monitoring.

c) Alarm and warning display

Displays alarm and warning status.

Display	Axis Status
No Alarm (Blue)	No alarm or warning occurs.
Alarm (Red)	An alarm is occurring.
Warning (Yellow)	A warning is occurring.

10

Troubleshooting

This chapter describes the causes of errors in motion programs and sequence programs, and suggests corrective actions.

10.1 Troubleshooting - - -

10.1 Troubleshooting

This section describes troubleshooting methods, and provides a list of motion program and sequence program errors.

10.1.1 Basic Flow of Troubleshooting

When a problem occurs, it is important to quickly find the cause of the problem and get the system running again as soon as possible. The basic troubleshooting flow is illustrated below.

Step 1	Visually confirm the following items.
• Machine movement (or status if stopped) • Power supply • I/O device status • Wiring status • Indicator status (LED indicators on each Module) • Switch settings (e.g., DIP switches) • Parameter settings and program contents	

↓

Step 2	Monitor the system to see if the problem changes in response to the following operations.
• Switching the Controller to STOP status • Resetting alarms • Turning the power supply OFF and ON again.	

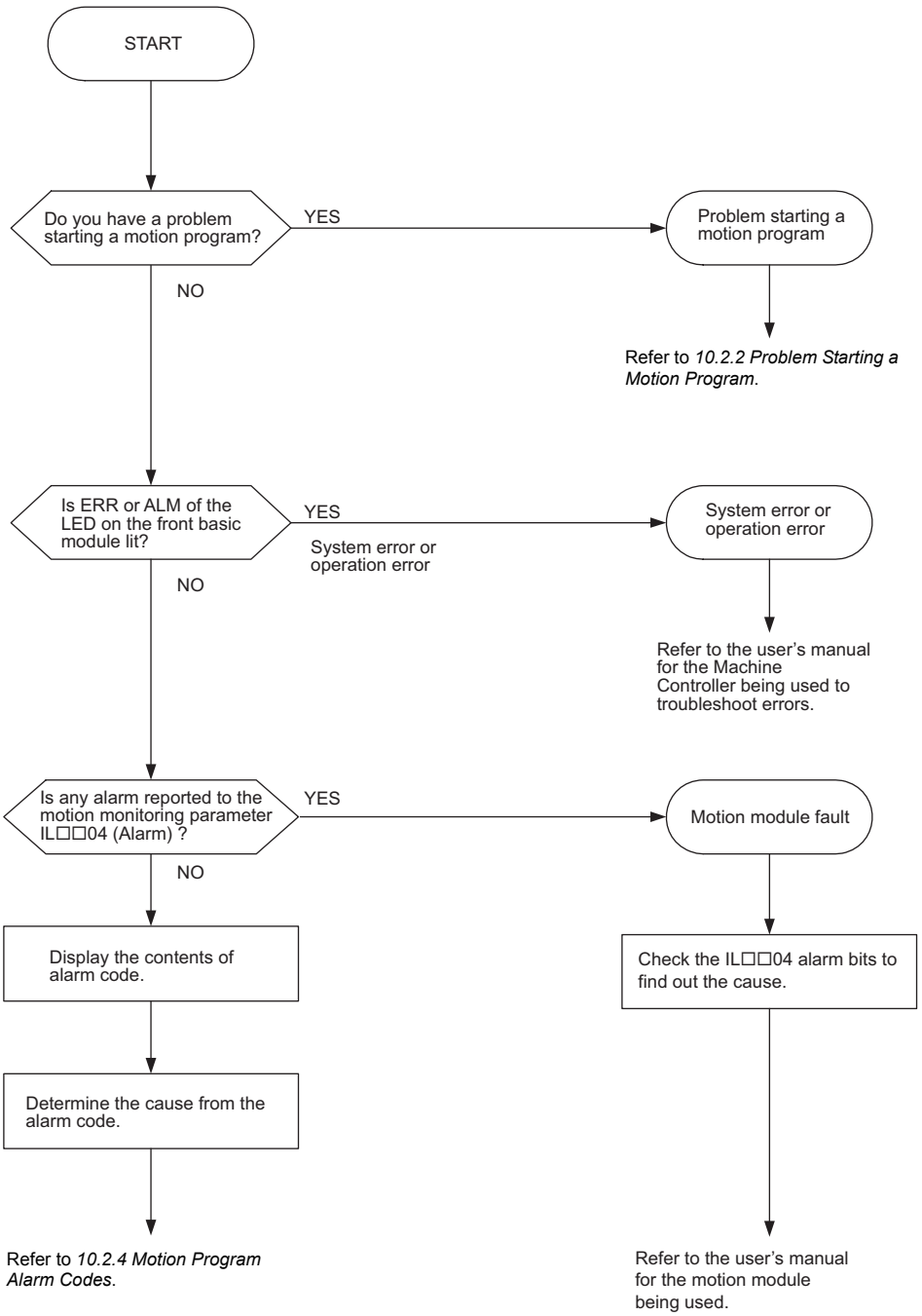
↓

Step 3	Determine the location of the cause from the results of steps 1 and 2.
• Controller or external? • Sequence control or motion control? • Software or hardware?	

10.2 Troubleshooting for Motion Programs

10.2.1 Error Investigation Flow

If the cause of error may be related to the motion program, refer to the following flowchart to troubleshoot.



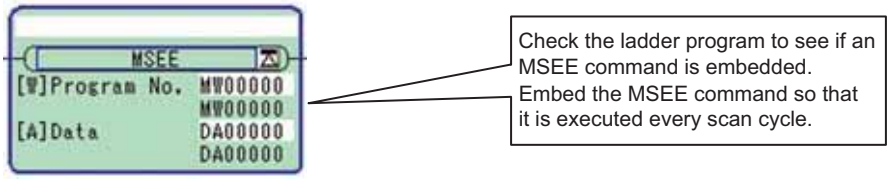
10.2.2 Problem Starting a Motion Program

If a problem exists when starting a motion program, check the following items to find out how to correct the problem.

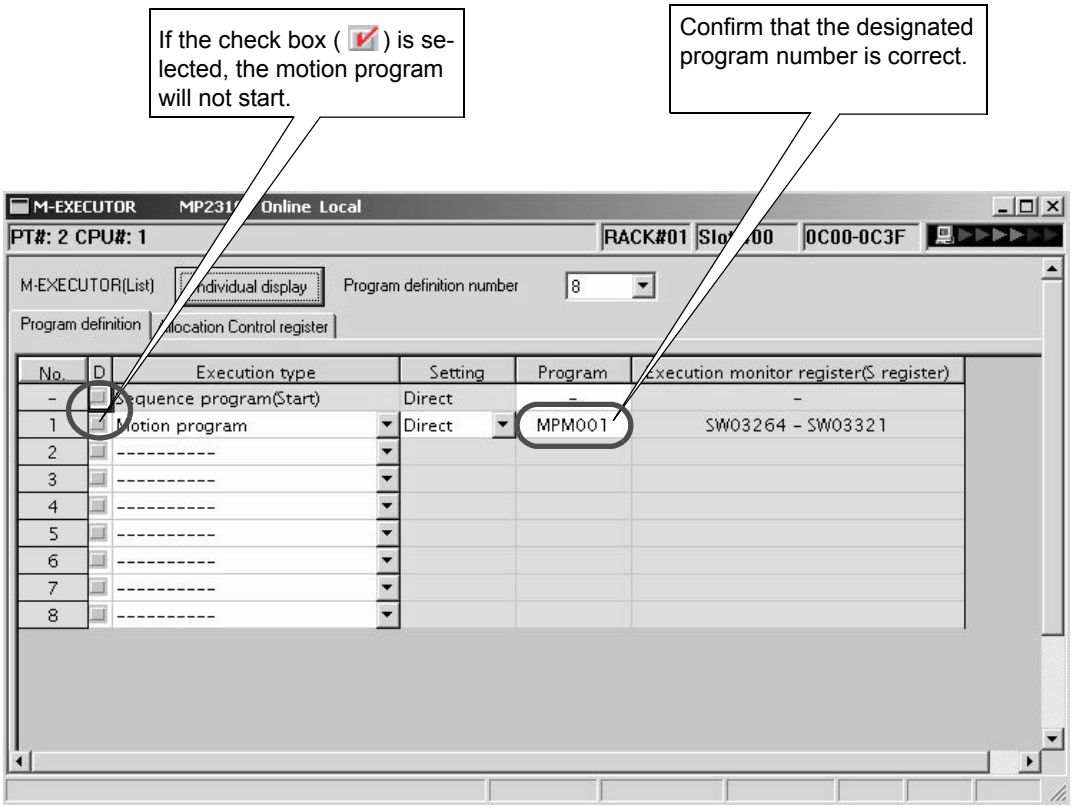
(1) Program Execution Registration to the System

Before starting the motion program, motion programs must be registered to the system used for execution. Two methods can be used to register motion programs: By embedding an MSEE command in H drawing or by registering motion programs to the M-EXECUTOR module.
Refer to *4.3.2 Registering the Program Execution* for details on motion program registration methods.

- Embedding an MSEE command in H Drawing



- Registering Motion Programs to M-EXECUTOR

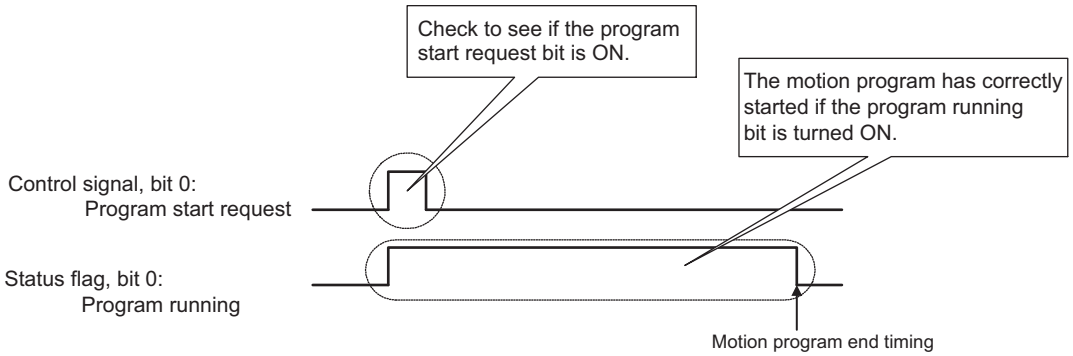


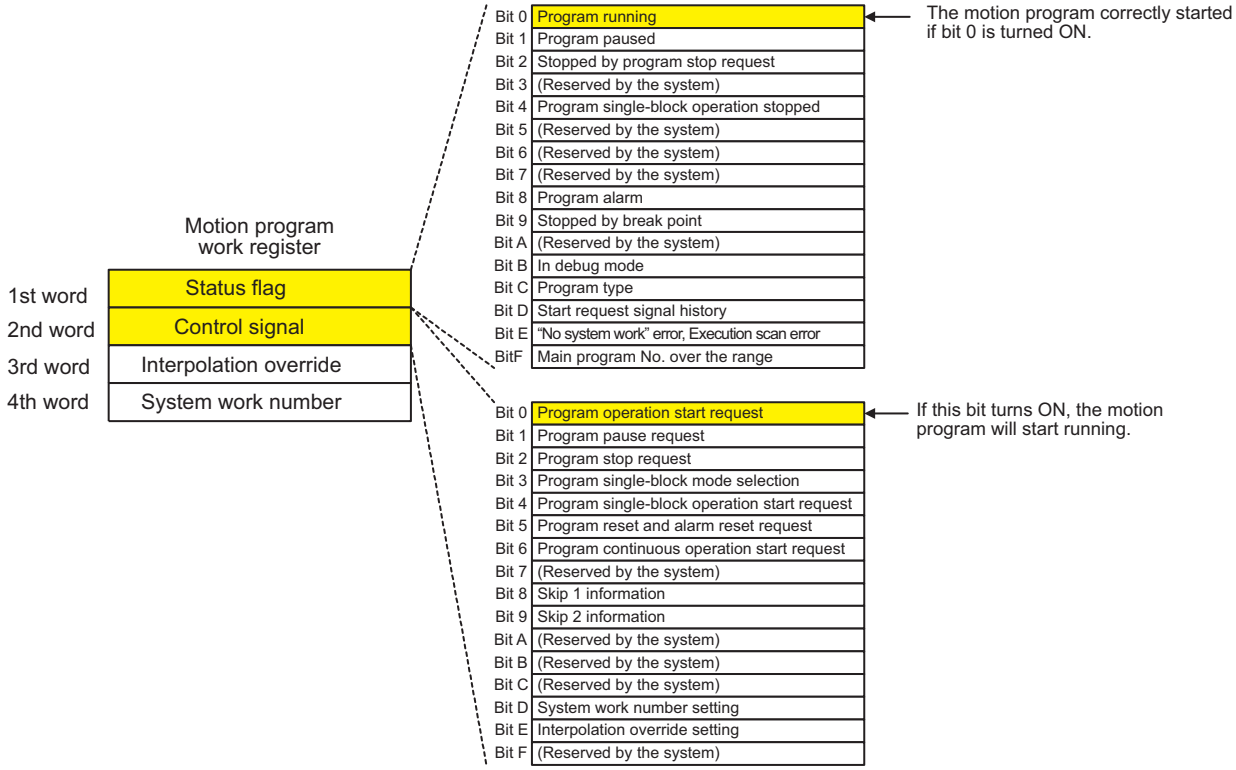
(2) Program Start Request Bit of Control Signal and Program Running Status Bit of Status Flag

When the MP2000 system detects a status change in motion program control signal bit 0 (Program start request) from OFF to ON, the motion program will start running. When the motion program successfully starts running, status flag signal bit 0 (Program running) will turn ON. Status flag signal bit 0 (Program running) will turn OFF when the END command in the motion program is executed.

To start the motion program again, turn motion program control signal bit 0 (Program start request) OFF, and then turn it ON again.

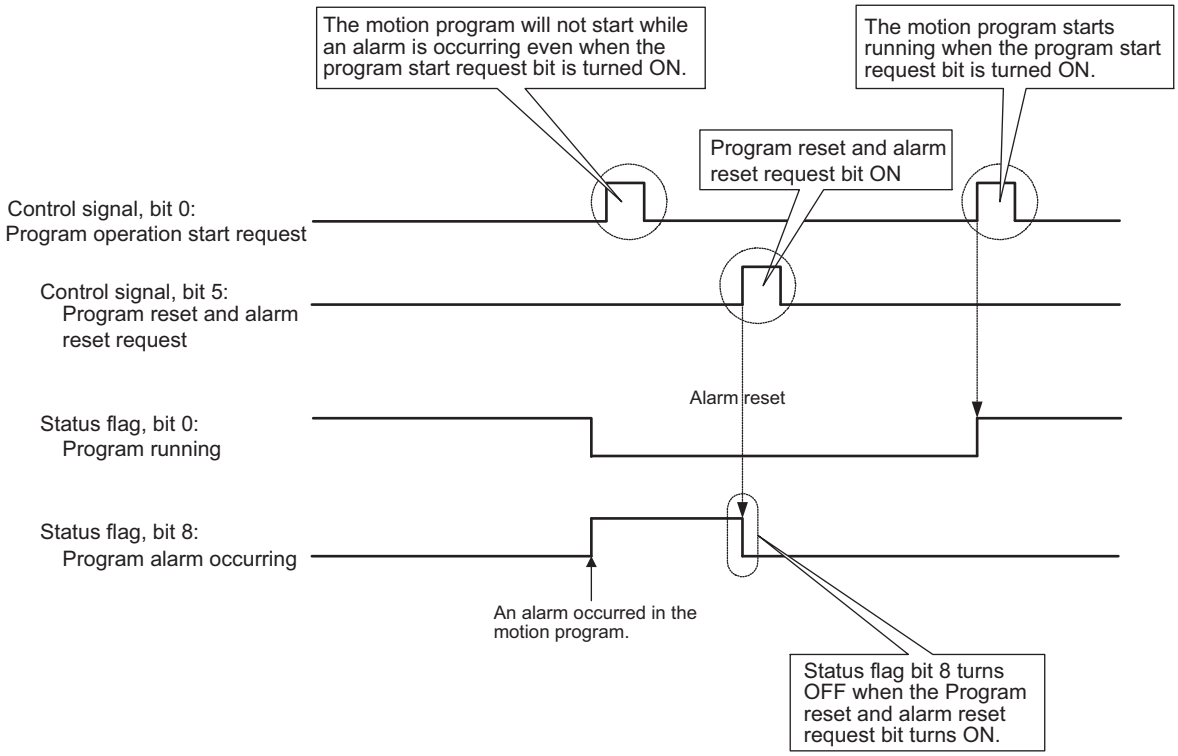
Check the ON/OFF status of these signal by using the data trace function.

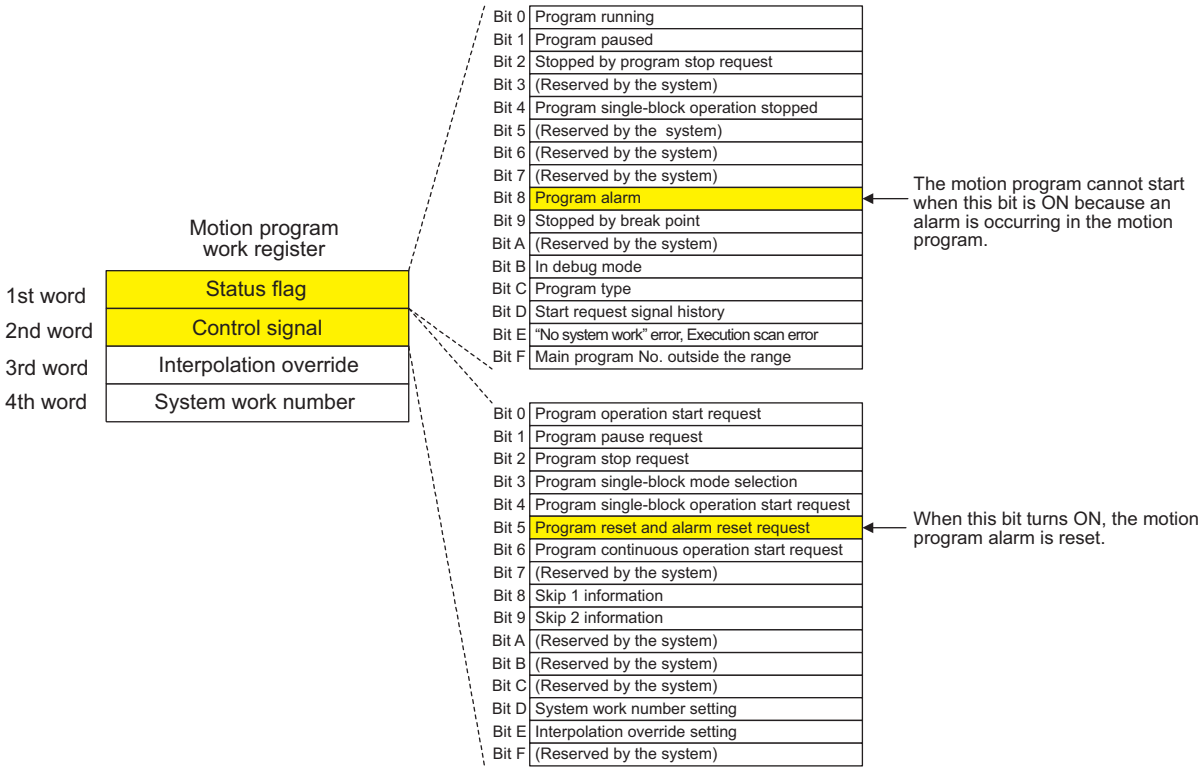




(3) Program Alarm Bit of Status Flag

If status flag bit 8 (Program alarm) is ON, the motion program is disabled from running because of the alarm in the motion program.
Remove the cause of the alarm by referring to *10.2.4 Motion Program Alarm Codes*, and then turn ON motion program control signal bit 5 (Program reset and alarm reset request) to reset the alarm. After the alarm is reset, start the motion program again.

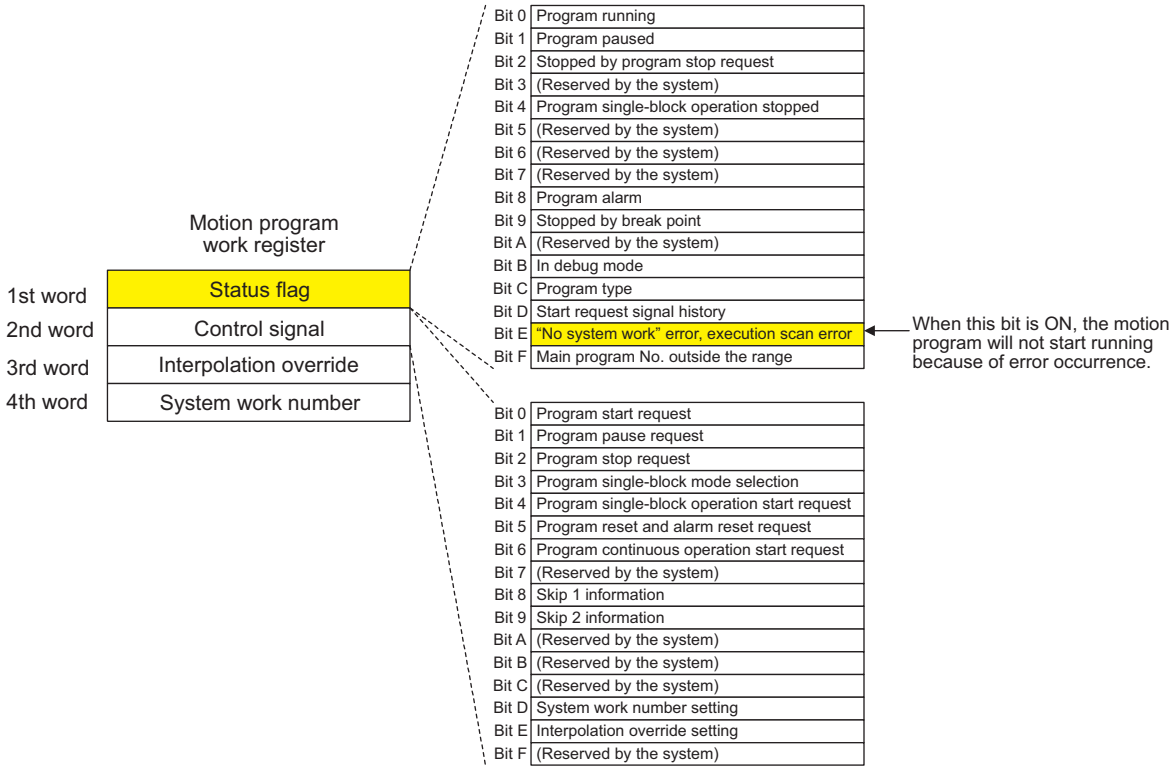




(4) "No System Work" Error/Execution Scan Error Bit of Status Flag

If running status flag bit E (“No system work” error/execution scan error) is ON, the motion program will be disabled from running because of error occurrence. In this case, confirm the following items:

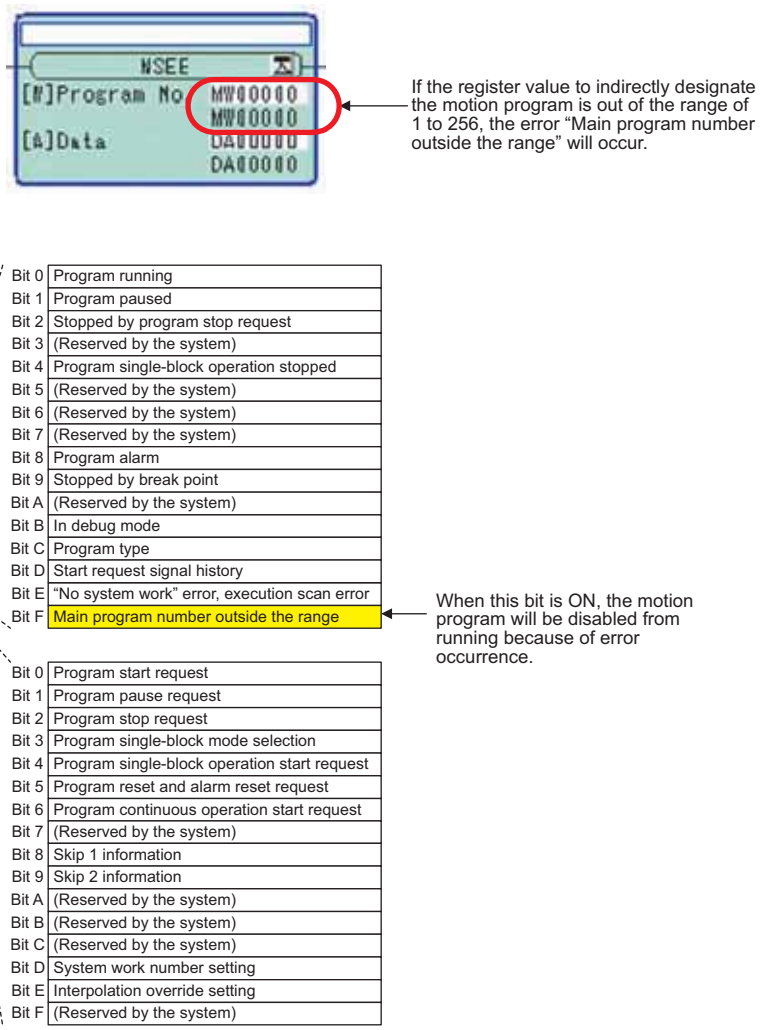
- The number of tasks being executed is less than 16.
- The task that the system work number designates is not being executed.
- The MSEE command is embedded in a high-speed scan drawing (DWG.H).



(5) Main Program Number Over the Range Status Flag Bit

When status flag bit F (Main program number outside the range) is ON, the motion program will be disabled from running because of an error. In this case, confirm the following:

- The motion program number designated in the MSEE command is within the range of 1 to 256.



10.2.3 Confirming the Alarm Code

When an alarm occurs in motion programs (status flag bit 8 [Program alarm] is ON), the alarm code indicates the cause of the alarm.
The alarm codes of motion programs can be found by using either of the following:

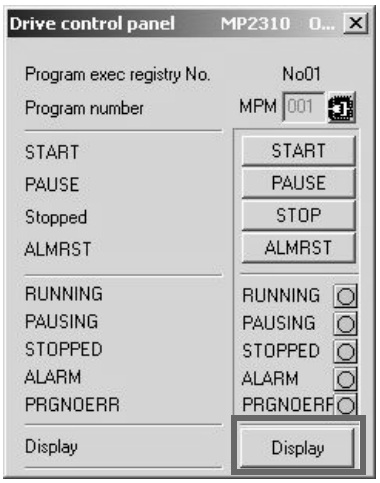
- **Error information screen** window
- S register

(1) Using Error Information Screen Window

There are two ways to open the **Error information screen** window.

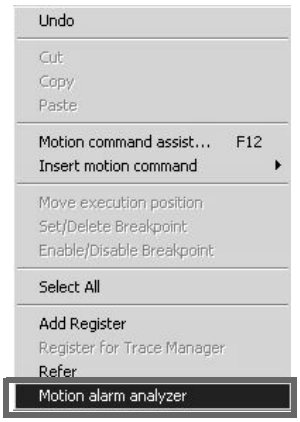
(a) Opening from the Drive Control Panel

Click the **Display** button on the **Drive control panel**.

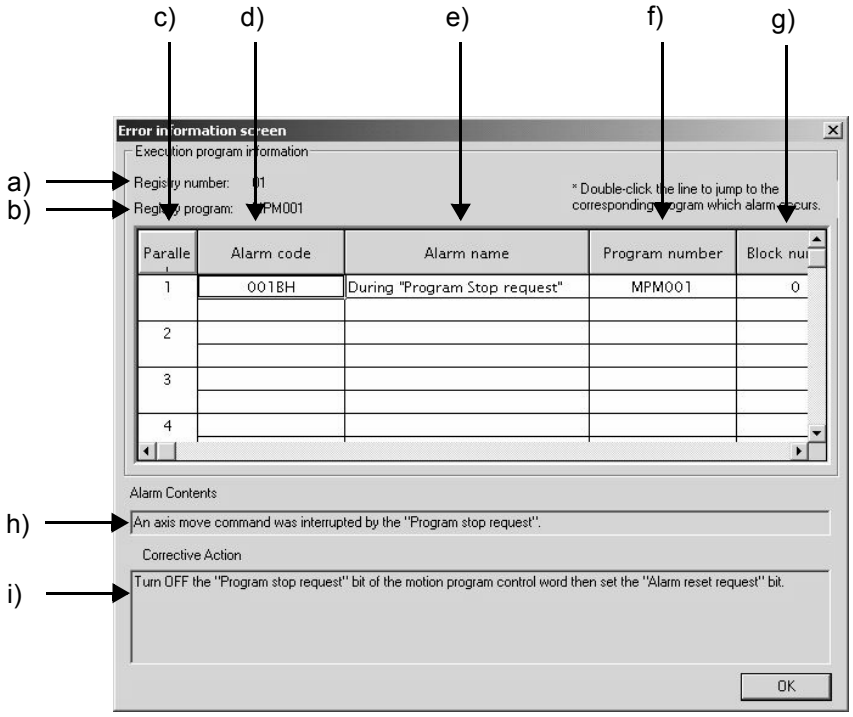


(b) Opening from the Pop-up Menu when Right-clicking on the Motion Editor Window

Right-click with the cursor on the **Motion Editor** window. Select **Motion alarm analyzer** from the menu that pops up.



The components of the **Error information screen** window are described below.



a) Registry number

When an alarm occurs in the motion program registered in the **Program Definition** window of the M-EXECUTOR, the registry number of the M-EXECUTOR defined in the **Program Definition** window is displayed.
When an alarm occurs in the motion program called from the ladder program by an MSEE command, "---" is displayed.

b) Registry program

When an alarm occurs in the motion program registered in the **Program Definition** window of the M-EXECUTOR, the motion program name defined in the **Program Definition** window is displayed.
When an alarm occurs in the motion program called from the ladder program by an MSEE command, "---" is displayed.

c) Parallel

When parallel execution commands (PFORK) are used in a motion program, more than one alarm may occur at once. Refer to *8.4.3 Parallel Execution (PFORK, JOINTO, PJOINT)* for details on parallel execution commands.

d) Alarm code

Alarm code is displayed.

e) Alarm name

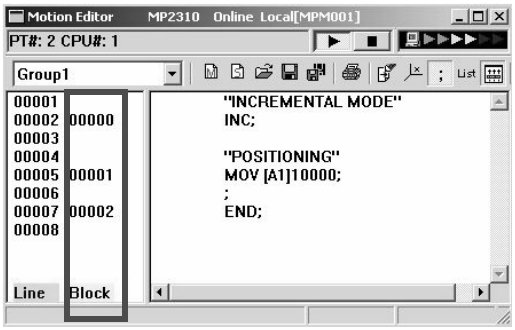
The alarm name is displayed.

f) Program number

The program number where an error occurs is displayed.

g) Block number

The block number where an error occurs is displayed.
Double-click the block number to jump to the corresponding program.
The block numbers are displayed on the **Motion Editor** window.



Block number

h) Alarm Contents

The contents of the alarm are displayed.

i) Corrective Action

The corrective actions for the alarm are displayed.

(2) Using S Register

The motion program alarm codes are stored in the motion program execution information of S registers (SW03200 to SW04191).
The S register number of the motion program alarm code will differ, depending on the system work to be used and the parallel number.
The motion program execution information of S registers (SW03200 to SW04191) is listed on the next page.

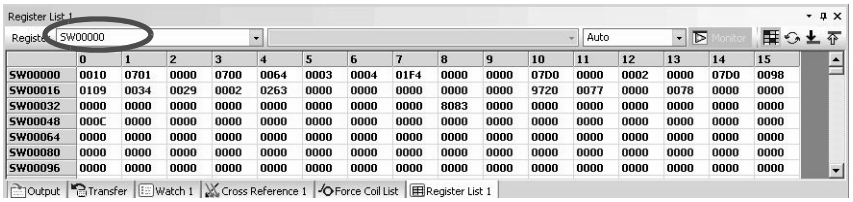
INFO

Use the following procedure to open the register list of MPE720 Ver.6.

1. Open the **Register List 1** subwindow from the MPE720 Ver.6 window.
   The **Register List 1** tab is provided on the bottom of the MPE720 Ver.6 window by default.



2. Enter the start number SW□□□□□ of system registers to view in the **Register** input field. The contents of system registers starting from the entered number will be displayed.



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
SW00000	0010	0701	0000	0700	0064	0003	0004	01F4	0000	0000	07D0	0000	0002	0000	07D0	0098
SW00016	0109	0034	0029	0002	0263	0000	0000	0000	0000	0000	9720	0077	0000	0078	0000	0000
SW00032	0000	0000	0000	0000	0000	0000	0000	0000	8083	0000	0000	0000	0000	0000	0000	0000
SW00048	000C	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
SW00064	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
SW00080	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
SW00096	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

Note: The default data type setting is decimal. The data type can be changed to hexadecimal: Right-click with the cursor on the list. Select **Hexadecimal** from the menu that pops up.

- System Work Number 1 to 8

System Work Number		System Work 1	System Work 2	System Work 3	System Work 4	System Work 5	System Work 6	System Work 7	System Work 8
Executing Main Program No.		SW03200	SW03201	SW03202	SW03203	SW03204	SW03205	SW03206	SW03207
Status		SW03264	SW03322	SW03380	SW03438	SW03496	SW03554	SW03612	SW3670
Control Signal		SW03265	SW03323	SW03381	SW03439	SW03497	SW03555	SW03613	SW3671
Parallel 0	Program Number	SW03266	SW03324	SW03382	SW03440	SW03498	SW03556	SW03614	SW3672
	Block Number	SW03267	SW03325	SW03383	SW03441	SW03499	SW03557	SW03615	SW3673
	Alarm Code	SW03268	SW03326	SW03384	SW03442	SW03500	SW03558	SW03616	SW3674
Parallel 1	Program Number	SW03269	SW03327	SW03385	SW03443	SW03501	SW03559	SW03617	SW3675
	Block Number	SW03270	SW03328	SW03386	SW03444	SW03502	SW03560	SW03618	SW3676
	Alarm Code	SW03271	SW03329	SW03387	SW03445	SW03503	SW03561	SW03619	SW3677
Parallel 2	Program Number	SW03272	SW03330	SW03388	SW03446	SW03504	SW03562	SW03620	SW3678
	Block Number	SW03273	SW03331	SW03389	SW03447	SW03505	SW03563	SW03621	SW3679
	Alarm Code	SW03274	SW03332	SW03390	SW03448	SW03506	SW03564	SW03622	SW3680
Parallel 3	Program Number	SW03275	SW03333	SW03391	SW03449	SW03507	SW03565	SW03623	SW3681
	Block Number	SW03276	SW03334	SW03392	SW03450	SW03508	SW03566	SW03624	SW3682
	Alarm Code	SW03277	SW03335	SW03393	SW03451	SW03509	SW03567	SW03625	SW3683
Parallel 4	Program Number	SW03278	SW03336	SW03394	SW03452	SW03510	SW03568	SW03626	SW3684
	Block Number	SW03279	SW03337	SW03395	SW03453	SW03511	SW03569	SW03627	SW3685
	Alarm Code	SW03280	SW03338	SW03396	SW03454	SW03512	SW03570	SW03628	SW3686
Parallel 5	Program Number	SW03281	SW03339	SW03397	SW03455	SW03513	SW03571	SW03629	SW3687
	Block Number	SW03282	SW03340	SW03398	SW03456	SW03514	SW03572	SW03630	SW3688
	Alarm Code	SW03283	SW03341	SW03399	SW03457	SW03515	SW03573	SW03631	SW3689
Parallel 6	Program Number	SW03284	SW03342	SW03400	SW03458	SW03516	SW03574	SW03632	SW3690
	Block Number	SW03285	SW03343	SW03401	SW03459	SW03517	SW03575	SW03633	SW3691
	Alarm Code	SW03286	SW03344	SW03402	SW03460	SW03518	SW03576	SW03634	SW3692
Parallel 7	Program Number	SW03287	SW03345	SW03403	SW03461	SW03519	SW03577	SW03635	SW3693
	Block Number	SW03288	SW03346	SW03404	SW03462	SW03520	SW03578	SW03636	SW3694
	Alarm Code	SW03289	SW03347	SW03405	SW03463	SW03521	SW03579	SW03637	SW3695
Logical axis #1 Program Current Position		SL03290	SL03348	SL03406	SL03464	SL03522	SL03580	SL03638	SL3696
Logical axis #2 Program Current Position		SL03292	SL03350	SL03408	SL03466	SL03524	SL03582	SL03640	SL3698
Logical axis #3 Program Current Position		SL03294	SL03352	SL03410	SL03468	SL03526	SL03584	SL03642	SL3700
Logical axis #4 Program Current Position		SL03296	SL03354	SL03412	SL03470	SL03528	SL03586	SL03644	SL3702
Logical axis #5 Program Current Position		SL03298	SL03356	SL03414	SL03472	SL03530	SL03588	SL03646	SL3704
Logical axis #6 Program Current Position		SL03300	SL03358	SL03416	SL03474	SL03532	SL03590	SL03648	SL3706
Logical axis #7 Program Current Position		SL03302	SL03360	SL03418	SL03476	SL03534	SL03592	SL03650	SL3708
Logical axis #8 Program Current Position		SL03304	SL03362	SL03420	SL03478	SL03536	SL03594	SL03652	SL3710
Logical axis #9 Program Current Position		SL03306	SL03364	SL03422	SL03480	SL03538	SL03596	SL03654	SL3712
Logical axis #10 Program Current Position		SL03308	SL03366	SL03424	SL03482	SL03540	SL03598	SL03656	SL3714
Logical axis #11 Program Current Position		SL03310	SL03368	SL03426	SL03484	SL03542	SL03600	SL03658	SL3716
Logical axis #12 Program Current Position		SL03312	SL03370	SL03428	SL03486	SL03544	SL03602	SL03660	SL3718

System Work Number	System Work 1	System Work 2	System Work 3	System Work 4	System Work 5	System Work 6	System Work 7	System Work 8
Logical axis #13 Program Current Position	SL03314	SL03372	SL03430	SL03488	SL03546	SL03604	SL03662	SL3720
Logical axis #14 Program Current Position	SL03316	SL03374	SL03432	SL03490	SL03548	SL03606	SL03664	SL3722
Logical axis #15 Program Current Position	SL03318	SL03376	SL03434	SL03492	SL03550	SL03608	SL03666	SL3724
Logical axis #16 Program Current Position	SL03320	SL03378	SL03436	SL03494	SL03552	SL03610	SL03668	SL3726

- System Word Number 9 to 16

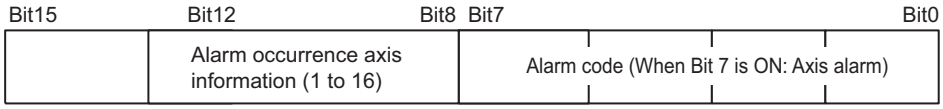
System Work Number		System Work 9	System Work 10	System Work 11	System Work 12	System Work 13	System Work 14	System Work 15	System Work 16
Executing Main Program No.		SW03208	SW03209	SW03210	SW03211	SW03212	SW03213	SW03214	SW03215
Status		SW03728	SW03786	SW03844	SW03902	SW03960	SW04018	SW04076	SW04134
Control Signal		SW03729	SW03787	SW03845	SW03903	SW03961	SW04019	SW04077	SW04135
Parallel 0	Program Number	SW03730	SW03788	SW03846	SW03904	SW03962	SW04020	SW04078	SW04136
	Block Number	SW03731	SW03789	SW03847	SW03905	SW03963	SW04021	SW04079	SW04137
	Alarm Code	SW03732	SW03790	SW03848	SW03906	SW03964	SW04022	SW04080	SW04138
Parallel 1	Program Number	SW03733	SW03791	SW03849	SW03907	SW03965	SW04023	SW04081	SW04139
	Block Number	SW03734	SW03792	SW03850	SW03908	SW03966	SW04024	SW04082	SW04140
	Alarm Code	SW03735	SW03793	SW03851	SW03909	SW03967	SW04025	SW04083	SW04141
Parallel 2	Program Number	SW03736	SW03794	SW03852	SW03910	SW03968	SW04026	SW04084	SW04142
	Block Number	SW03737	SW03795	SW03853	SW03911	SW03969	SW04027	SW04085	SW04143
	Alarm Code	SW03738	SW03796	SW03854	SW03912	SW03970	SW04028	SW04086	SW04144
Parallel 3	Program Number	SW03739	SW03797	SW03855	SW03913	SW03971	SW04029	SW04087	SW04145
	Block Number	SW03740	SW03798	SW03856	SW03914	SW03972	SW04030	SW04088	SW04146
	Alarm Code	SW03741	SW03799	SW03857	SW03915	SW03973	SW04031	SW04089	SW04147
Parallel 4	Program Number	SW03742	SW03800	SW03858	SW03916	SW03974	SW04032	SW04090	SW04148
	Block Number	SW03743	SW03801	SW03859	SW03917	SW03975	SW04033	SW04091	SW04149
	Alarm Code	SW03744	SW03802	SW03860	SW03918	SW03976	SW04034	SW04092	SW04150
Parallel 5	Program Number	SW03745	SW03803	SW03861	SW03919	SW03977	SW04035	SW04093	SW04151
	Block Number	SW03746	SW03804	SW03862	SW03920	SW03978	SW04036	SW04094	SW04152
	Alarm Code	SW03747	SW03805	SW03863	SW03921	SW03979	SW04037	SW04095	SW04153
Parallel 6	Program Number	SW03748	SW03806	SW03864	SW03922	SW03980	SW04038	SW04096	SW04154
	Block Number	SW03749	SW03807	SW03865	SW03923	SW03981	SW04039	SW04097	SW04155
	Alarm Code	SW03750	SW03808	SW03866	SW03924	SW03982	SW04040	SW04098	SW04156
Parallel 7	Program Number	SW03751	SW03809	SW03867	SW03925	SW03983	SW04041	SW04099	SW04157
	Block Number	SW03752	SW03810	SW03868	SW03926	SW03984	SW04042	SW04100	SW04158
	Alarm Code	SW03753	SW03811	SW03869	SW03927	SW03985	SW04043	SW04101	SW04159
Logical axis #1 Program Current Position		SL03754	SL03812	SL03870	SL03928	SL03986	SL04044	SL04102	SL04160
Logical axis #2 Program Current Position		SL03756	SL03814	SL03872	SL03930	SL03988	SL04046	SL04104	SL04162
Logical axis #3 Program Current Position		SL03758	SL03816	SL03874	SL03932	SL03990	SL04048	SL04106	SL04164
Logical axis #4 Program Current Position		SL03760	SL03818	SL03876	SL03934	SL03992	SL04050	SL04108	SL04166
Logical axis #5 Program Current Position		SL03762	SL03820	SL03878	SL03936	SL03994	SL04052	SL04110	SL04168
Logical axis #6 Program Current Position		SL03764	SL03822	SL03880	SL03938	SL03996	SL04054	SL04112	SL04170
Logical axis #7 Program Current Position		SL03766	SL03824	SL03882	SL03940	SL03998	SL04056	SL04114	SL04172

System Work Number	System Work 9	System Work 10	System Work 11	System Work 12	System Work 13	System Work 14	System Work 15	System Work 16
Logical axis #8 Program Current Position	SL03768	SL03826	SL03884	SL03942	SL04000	SL04058	SL04116	SL04174
Logical axis #9 Program Current Position	SL03770	SL03828	SL03886	SL03944	SL04002	SL04060	SL04118	SL04176
Logical axis #10 Program Current Position	SL03772	SL03830	SL03888	SL03946	SL04004	SL04062	SL04120	SL04178
Logical axis #11 Program Current Position	SL03774	SL03832	SL03890	SL03948	SL04006	SL04064	SL04122	SL04180
Logical axis #12 Program Current Position	SL03776	SL03834	SL03892	SL03950	SL04008	SL04066	SL04124	SL04182
Logical axis #13 Program Current Position	SL03778	SL03836	SL03894	SL03952	SL04010	SL04068	SL04126	SL04184
Logical axis #14 Program Current Position	SL03780	SL03838	SL03896	SL03954	SL04012	SL04070	SL04128	SL04186
Logical axis #15 Program Current Position	SL03782	SL03840	SL03898	SL03956	SL04014	SL04072	SL04130	SL04188
Logical axis #16 Program Current Position	SL03784	SL03842	SL03900	SL03958	SL04016	SL04074	SL04132	SL04190

10.2.4 Motion Program Alarm Codes

(1) Configuration of Motion Program Alarms

The following diagram shows the configuration of alarms.



(2) Alarm Code List for Motion Program

The following table shows the alarm codes of motion programs.

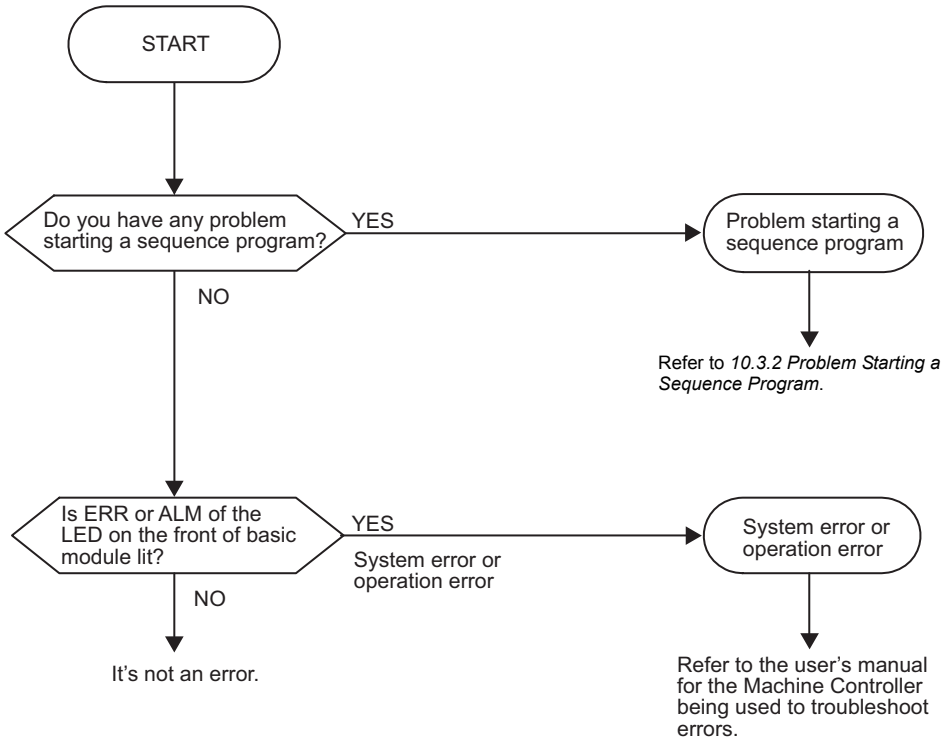
Alarm Code	Name	Description	Corrective Actions
02h	Division error	Data divided by 0	Review the motion program.
10h	A circle instead of radius was specified	Turn number was specified instead of radius in the circular arc or helical interpolation command.	• Designate a center coordinate instead of a radius to perform the circular arc or helical interpolation command. • Never specify the turn number.
11h	Interpolation feeding speed over limit	Interpolation feeding speed exceeded the valid range of the FMX command.	Modify the interpolation feeding speed of the interpolation command
12h	No interpolation feeding speed specified	No interpolation feeding speed was specified. (once specified, this can be omitted as in the motion program)	Specify the interpolation feeding speed in the interpolation command.
13h	Range exceeded after converting acceleration parameter	Indirect acceleration parameter exceeded the valid range.	Change the indirect register value.
14h	Circular arc length exceeded LONG_MAX	Circular arc length exceeded the valid range in the circular arc or helical interpolation command.	Review the circular arc length in the circular arc or helical interpolation command.
15h	Vertical axis not specified for circular arc plane	Vertical axis was not specified in the circular arc or helical interpolation command.	Use PLN command to specify the axis.
16h	Horizontal axis not specified for circular arc plane	Horizontal axis was not specified in the circular arc or helical interpolation command.	Use PLN command to specify the axis.
17h	Specified axis over limit	Too many axes were configured in the circular arc (two axes) or helical (three axes) interpolation command.	Modify the axis in the circular arc or helical interpolation command.
18h	Turn number over limit	Turn number exceeded the valid range in the circular arc or helical interpolation command.	Modify the turn number in the circular arc or helical interpolation command.
19h	Radius exceeded LONG_MAX	Radius exceeded the valid range in the circular arc or helical interpolation command.	Review the radius in the circular arc or helical interpolation command.
1Ah	Center point error	Improper center point was specified in the circular arc or helical interpolation command.	Specify the center point properly in the circular arc or helical interpolation command.
1Bh	Running emergency stop command	Axis move command stopped due to a program stop request.	Turn OFF the program stop request for the motion program control signal, and turn ON the alarm reset request.
1Ch	Linear interpolation moving amount exceeded LONG_MAX	Moving amount exceeded the valid range in the linear interpolation command.	Review the moving amount in the linear interpolation command.
1Dh	FMX undefined	FMX command not executed in the motion program containing an interpolation command.	Perform an FMX command. The FMX command is required in each program containing an interpolation command.
1Eh	Address T out of range	Designation exceeded the valid range in the IAC/IDC/FMX commands.	Review the setting in the IAC/IDC/FMX command.

Alarm Code	Name	Description	Corrective Actions
1Fh	Address P out of range	Designation exceeded the valid range in the IFP command.	Review the setting in the IFP command.
21h	PFORK execution error	A motion command was instructed simultaneously at the second line in the PFORK of both a source motion program and a subprogram.	Review the source motion program or subprogram.
22h	Indirect register range error	Specified register address exceeds the register size range.	Review the motion program.
23h	Moving amount out of range	Axis moving amount with decimal point for an axis move command exceeded the possible range.	Review the axis moving amount.
80h	Use of logical axis prohibited	Multiple motion commands instructed against the same axis at the same time.	Review the motion program.
81h	Designation exceeded POSMAX in the infinite length axis	Moving distance designation exceeded POSMAX in the infinite length axis.	• Modify the fixed parameter “Maximum infinite length axis counter” • Review the motion program.
82h	Axis moving distance exceeded LONG_MAX	Axis moving distance designation exceeded the valid range.	Review the motion program.
84h	Duplicated motion command	Multiple commands ware executed against a single axis.	Check to see whether another program gave a command to the same axis at the same time. If so, review the program.
85h	Motion command response error	A motion command response different from that instructed by the motion command is reported from a motion module.	• Remove the alarm cause from the destination axis. • If the servo is not turned ON, turn ON the servo. • Check to see whether another program gave a command to the same axis at the same time. If so, review the program.
87h	VEL setting data out of range	An instruction in the VEL command exceeded the valid range.	Review the VEL command.
88h	INP setting data out of range	An instruction in the INP command exceeded the valid range.	Review the INP command.
89h	ACC/SCC/DCC setting data out of range	An instruction in the ACC/SCC/DCC command exceeded the valid range.	Review the ACC/SCC/DCC command.
8Ah	No time specified in the MVT command	T designation in the MVT command was zero.	Review the MVT command.
8Bh	Command execution disabled	A motion command which cannot be executed by the destination motion module was instructed.	Review the motion program.
8Ch	Distribution incomplete	A motion command was executed when a motion module was not in the Distribution Completed state.	Review the motion program so that the motion command is executed in the Distribution Completed state.
8Dh	Motion command abnormally aborted	Motion module fell into the “Motion command abnormally aborted” state.	• Release the destination axis error. • Review the motion program.

10.3 Troubleshooting for Sequence Programs

10.3.1 Error Investigation Flow

If the cause of error may be related to the sequence program, refer to the following flowchart to troubleshoot the error.

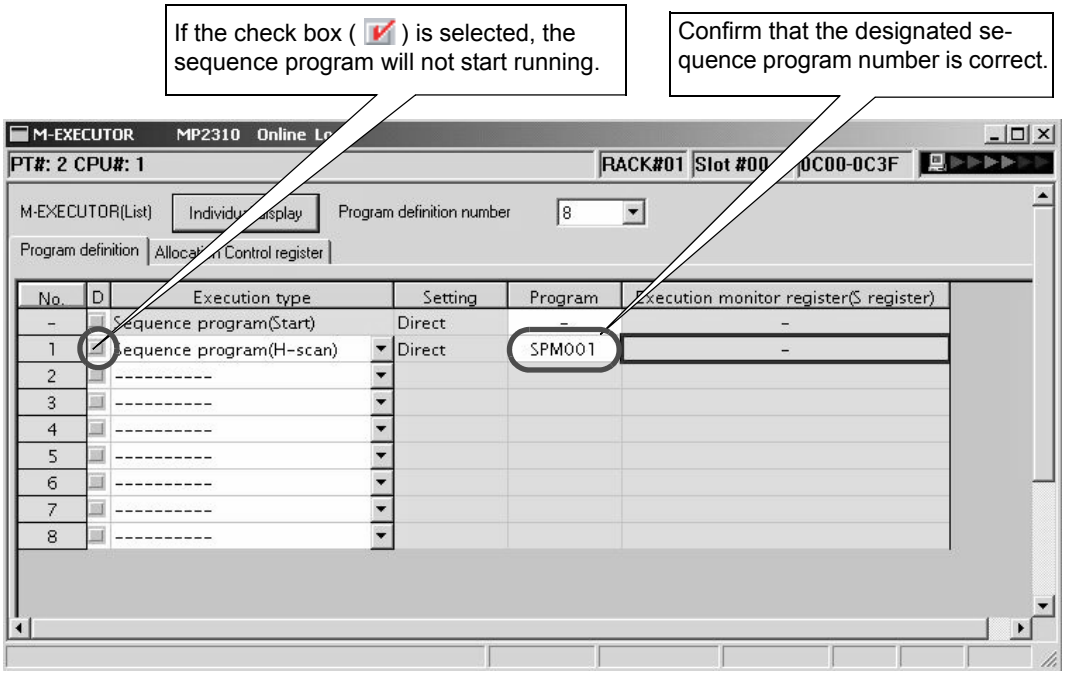


10.3.2 Problem Starting a Sequence Program

When a problem exists when starting a sequence program, check the following items to find out how to correct it.

(1) Program Execution Registration to the System

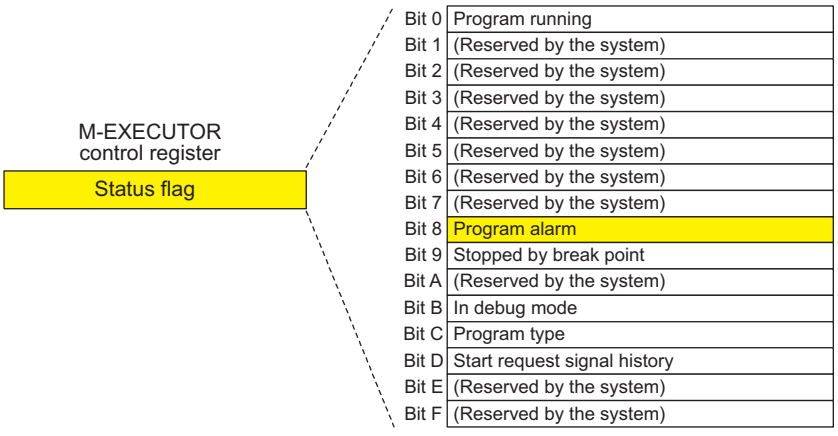
Before starting the sequence program, sequence programs must be registered to the system used to execute. Register the sequence programs to the M-EXECUTOR module to register them to the system for execution. Refer to *5.2.2 Registering Program Execution* for information for sequence program registration.



(2) Program Alarm Occurring Status Flag Bit

When status flag bit 8 (Program alarm occurring) is ON, the sequence program is disabled from running because of an error occurrence. In this case, check the following items:

- The called program exists.
- The called program is a sequence program.
- The sequence subprogram call command (SSEE) calls a subprogram, but not a main program.
- The sequence program number designated in the SSEE command is within the range of 1 to 256.
- Nesting of sequence subprogram call command (SSEE) is within 8 levels.



Appendices

A Motion Language Commands ---------- A-2
A.1 Axis Setting Commands ---------- A-2
A.2 Axis Move Commands ---------- A-3
A.3 Control Commands ---------- A-5
A.4 Program Control Commands ---------- A-6
A.5 Arithmetic Operations ---------- A-8
A.6 Logical Operations ---------- A-8
A.7 Data Comparison ---------- A-9
A.8 Data Operations ---------- A-9
A.9 Basic Functions ---------- A-10
A.10 C-Language Control Commands ---------- A-11
B Sample Programs ---------- A-12
B.1 Programs for Controlling Motion Program Execution ---------- A-13
B.2 Parallel Processing ---------- A-15
B.3 Motion Program for Speed Control ---------- A-16
B.4 Simple Synchronized Operation Using a Virtual Axis ---------- A-17
B.5 Sequence Programs ---------- A-19
C Differences between MP900 Series and MP2000 Series Machine Controllers ---------- A-21
C.1 Motion Programs ---------- A-21
C.2 Sequence Programs ---------- A-21
C.3 Motion Programming Commands ---------- A-22
C.4 Group Definitions ---------- A-22
C.5 Debug Function ---------- A-23
C.6 Motion Program Alarms ---------- A-23
D Precautions ---------- A-24
D.1 General Precautions ---------- A-24
D.2 Precautions on Motion Parameter Settings ---------- A-24

App

A Motion Language Commands

Motion language commands are listed below. Refer to *Chapter 8 Command Reference* for details on each command.

✓	–
Applicable	Not applicable

A.1 Axis Setting Commands

Command	Name	Programming Format	Description	Motion Programs	Sequence Programs
ABS	Absolute Mode	ABS; or ABS MOV [Logical axis name 1] - [Logical axis name 2] - ;	Treats all subsequent coordinate words as absolute values.	✓	–
INC	Incremental Mode	INC; or INC MOV [Logical axis name 1] - [Logical axis name 2] - ;	Treats all subsequent coordinate words as incremental values.	✓	–
ACC	Acceleration Time Change	ACC [Logical axis name 1]Acceleration time [Logical axis name 2] Acceleration time [Logical axis name 3] Acceleration time ⋯ ;	Sets the acceleration time for linear acceleration/deceleration for up to 16 axes simultaneously.	✓	–
DCC	Deceleration Time Change	DCC [Logical axis name 1] Deceleration time [Logical axis name 2] Deceleration time [Logical axis name 3] Deceleration time ⋯ ;	Sets the deceleration time for a positioning related command. A maximum of 16 axes can be designated in one command block.	✓	–
SCC	S-curve Time Constant Change	SCC [Logical axis name 1] S-curve time constant [Logical axis name 2] S-curve time constant ⋯ ;	Sets the time constant for moving average acceleration/deceleration for up to 16 axes simultaneously.	✓	–
VEL	Set Velocity	VEL [Logical axis name 1] Feed speed [Logical axis name 2] Feed speed [Logical axis name 3] Feed speed ⋯ ;	Sets the speed for a positioning related command. A maximum of 16 axes can be designated in one command block.	✓	–
FMX	Maximum Interpolation Feed Speed Setting	FMX T Maximum interpolation feed speed ;	Sets the maximum speed during an interpolation feed. The interpolation acceleration time is the time taken from “0” until this speed is reached.	✓	–
IFP	Interpolation Feed Speed Ratio Setting	IFP P Interpolation feed speed ratio ;	Sets the speed for an interpolation related command. Designate a percentage of the maximum speed as the interpolation feed speed ratio.	✓	–
IAC	Interpolation Acceleration Time Change	IAC T Interpolation acceleration time ;	Sets the acceleration time for an interpolation related command. Designate the time required to reach the maximum speed from the speed 0 (zero).	✓	–
IDC	Interpolation Deceleration Time Change	IDC T Interpolation deceleration time ;	Sets the deceleration time for an interpolation related command. Designate the time required to decelerate to the speed 0 (zero) from the maximum speed.	✓	–

A.2 Axis Move Commands

Command	Name	Programming Format		Description	Motion Programs	Sequence Programs
MOV	Positioning	MOV [Logical axis name 1] Reference position [Logical axis name 2] Reference position [Logical axis name 3] Reference position ··· ;		Executes positioning at the specified positioning speed for up to 16 axes.	✓	–
MVS	Linear Interpolation	MVS [Logical axis name 1] Reference position [Logical axis name 2] Reference position [Logical axis name 3] Reference position ··· F Interpolation feed speed ;		Executes linear movement at the interpolation feed speed F for up to 16 axes.	✓	–
MCW	Clockwise Circular Interpolation	Center position designation	MCW [Logical axis name 1] End position [Logical axis name 2] End position U Center point position V Center point position T Number of turns F Interpolation speed ;	Executes circular interpolation at tangential speed F for two axes simultaneously following radius R (or designated center point coordinates). With the center point coordinate designation, multiple circles can be designated with T–. (T– can also be omitted.)	✓	–
		Radius designation	MCW [Logical axis name 1] End position [Logical axis name 2] End position R Radius F Interpolation feed speed ;			
MCC	Counterclockwise Circular Interpolation	Center position designation	MCC [Logical axis name 1] End position [Logical axis name 2] End position U Center point position V Center point position T Number of turns F Interpolation feed speed ;			
		Radius designation	MCC [Logical axis name 1] End position [Logical axis name 2] End position R Radius F Interpolation feed speed ;			
MCW	Clockwise Helical Interpolation	Center position designation	MCW [Logical axis name 1] End position [Logical axis name 2] End position U Center point position V Center point position [Logical axis name 3] Linear interpolation end position T Number of turns F Interpolation feed speed ;	Moves three axes simultaneously in a combination of circular interpolation and linear interpolation outside the circular interpolation plane. Speed F will be the circular interpolation tangential speed. With the center point coordinate designation, the number of turns can be designated with T–. (T– can also be omitted.)	✓	–
		Radius designation	MCW [Logical axis name 1] End position [Logical axis name 2] End position R Radius [Logical axis name 3] Linear interpolation end position F Interpolation feed speed ;			
MCC	Counterclockwise Helical Interpolation	Center position designation	MCC [Logical axis name 1] End position [Logical axis name 2] End position U Center point position V Center point position [Logical axis name 3] Linear interpolation end position T Number of turns F Interpolation feed speed ;			
		Radius designation	MCC [Logical axis name 1] End position [Logical axis name 2] End position R Radius [Logical axis name 3] Linear interpolation end position F Interpolation feed speed ··· ;			

- Continues to the next page -

(cont'd)

Com-mand	Name	Programming Format	Description	Motion Programs	Sequence Programs
ZRN	Zero Point Return	ZRN [Logical axis name1]0 [Logical axis name 2]0 [Logical axis name 3] ;	Returns each axis to its zero point.	✓	–
SKP	Linear Interpola-tion with Skip Function	SKP [Logical axis name 1] Reference position [Logical axis name 2] Reference position [Logical axis name 3] Reference position ··· F Interpolation feed speed SS Skip input signal ;	If the SKIP signal turns ON during a linear interpolation operation, skips the remaining movement and pro-ceeds to the next block.	✓	–
MVT	Set Time Positioning	MVT [Logical axis name 1] Reference position [Logical axis name 2] Reference position [Logical axis name 3] Reference position ··· T Positioning time (ms) ;	Executes positioning by clamping the feed speed so that travel can be com-pleted at the designated time.	✓	–
EXM	External Positioning	EXM [Logical axis name 1] Reference position D Movement amount from the time the external posi-tioning signal is input ;	When an external positioning signal is input while external positioning is being executed, only the travel dis-tance designated by "D–" is posi-tioned with an incremental value, and then the next command is executed.	✓	–

A.3 Control Commands

Command	Name	Programming Format	Description	Motion Programs	Sequence Programs
POS	Current Position Set	POS [Logical axis name 1] Desired coordinate value [Logical axis name 2] Desired coordinate value … ;	Changes the current values to the desired coordinate values for up to 16 axes. Subsequent move commands use this new coordinate system.	✓	–
MVM	Move on Machine Coordinate	MVM MOV [Logical axis name 1] Reference position [Logical axis name 2] Reference position [Logical axis name 3] Reference position … ;	Goes to the target position on the machine coordinate system. The coordinate system set automatically on completion of the zero point return is called a machine coordinate system. This coordinate system is not affected by the POS command.	✓	–
PLN	Coordinate Plane Setting	PLN [Logical axis name 1 (vertical axis)] [Logical axis name 2 (horizontal axis)] ;	Designates the coordinate plane to be used for a command requiring a plane designation command.	✓	–
PLD	Program Current Position Update	PLD [Logical axis name 1] [Logical axis name 2] … ;	Updates the program current position for axes shifted by manual intervention. Up to 16 axes can be designated.	✓	–
PFN	In-position Check	MVS [Logical axis name 1] - [Logical axis name 2] - … PFN ; or MVS [Logical axis name 1] - [Logical axis name 2] - … ; PFN [Logical axis name 1] [Logical axis name 2] ; MVS [Logical axis name 1] - [Logical axis name 2] - … ;	Proceeds to the next block when the axis that is being moved by an interpolation related command in the same block or the previous block, enters the in-position check width.	✓	–
INP	Set In-position Check Width	INP [Logical axis name 1] NEAR signal output width [Logical axis name 2] NEAR signal output width … ;	Sets the NEAR signal output width. The execution of a subsequent interpolation related command block with a PFN command proceeds to the next block after entering in-position check width.	✓	–

A.4 Program Control Commands

Com-mand	Name	Programming Format	Description	Motion Programs	Sequence Programs
IF ELSE IEND	Branching Commands	IF (Conditional expression) ; (Process 1) ; ELSE; (Process 2) ; IEND;	Executes process 1 if the conditional expression is satisfied, and executes process 2 if the conditional expression is not satisfied.	✓	✓
WHILE WEND	Repeat Commands	WHILE (Conditional expression) ; … ; WEND;	Repeatedly executes WHILE to WEND processing for as long as the conditional expression is satisfied.	✓	✓
PFORK JOINTO PJOINT	Parallel Execution Commands	PFORK Label 1, Label 2, Label 3, … ; Label 1: Process 1 ; JOINTO Label X ; Label 2: Process 2 ; JOINTO Label X ; Label 3: Process 3 ; JOINTO Label X ; Label X: PJOINT ;	Executes the blocks designated by the labels in parallel. With a subroutine, a maximum of two labels can be designated. END and RET cannot be used during parallel execution processing.	✓	–
SFORK JOINTO SJOINT	Selective Execution Commands	SFORK Conditional expression 1? Label 1, Conditional expression 2? Label 2, Conditional expression 3? Label 3, Conditional expression 4? Label 4 ; Label 1: Process 1 ; JOINTO Label X ; Label 2: Process 2 ; JOINTO Label X ; Label 3: Process 3 ; JOINTO Label X ; Label 4: Process 4 ; JOINTO Label X ; … ; *Label X*: SJOINT;	Executes process 1 if conditional expression 1 is satisfied, and executes process 2 if the conditional expression 2 is satisfied.	✓	✓
MSEE	Motion Subprogram Call	MSEE MPS□□□ ;	Executes subprogram MPS□□□.	✓	–
SSEE	Sequence Subprogram Call	SSEE SPS□□□ ;	Executes subprogram SPS□□□.	–	✓
UFC	User Function Call From Motion Program	UFC User function name Input data, Input address, Output data ;	Calls the user defined function from a motion program.	✓	–
FUNC	User Function Call From Sequence Program	FUNC User function name, Input data, Input address, Output data ;	Calls the user defined function from a sequence program.	–	✓
END	Program End	END;	Ends the motion program.	✓	✓
RET	Subroutine Return	RET;	Ends the subroutine.	✓	✓

(cont'd)

Com-mand	Name	Programming Format	Description	Motion Programs	Sequence Programs
TIM	Dwell Time	TIM T–	Waits for the period of time specified by T, and then proceeds to the next block.	✓	–
IOW	I/O Wait	IOW MB–== ··· ;	Stops execution of the motion program until the conditional expression given in the command is satisfied.	✓	–
EOX	One Scan Wait	EOX;	Divides the execution of consecutively coded sequence commands. The command block after EOX will be executed at the next scan.	✓	–
SNGD/ SNGE	Single-block Disabled (SNGD)/Single-block Enabled (SNGE)	SNGD; ··· ; SNGE;	Specifies whether to enable or disable single step operation during debugging.	✓	–

A.5 Arithmetic Operations

Command	Name	Programming Format	Description	Motion Programs	Sequence Programs
=	Substitute	(Result) = (Math expression)	Substitutes operation results. Performs calculations from left to right (with no order of priority).	✓	✓
+	Add	MW – = MW – + MW – ;	Performs integer and real number addition. Calculates combinations of integers and real numbers as real numbers.	✓	✓
-	Subtract	MW – = MW – - MW – ;	Performs integer and real number subtraction. Calculates combinations of integers and real numbers as real numbers.	✓	✓
*	Multiply	MW – = MW – * MW – ;	Performs integer and real number multiplication. Calculates combinations of integers and real numbers as real numbers.	✓	✓
/	Divide	MW – = MW –/ MW – ;	Performs integer and real number division. Calculates combinations of integers and real numbers as real numbers.	✓	✓
MOD	Remainder	MW – = MW –/ MW – ; MW – = MOD;	When programmed in the next block after a division, MOD stores the remainder in the designated register.	✓	✓

A.6 Logical Operations

Command	Name	Programming Format	Description	Motion Programs	Sequence Programs
|	OR (logical OR)	MB – = MB – | MB – ; MB – = MB – | 1; MW – = MW – | MW – ; MW – = MW – | 00FFH;	Performs bit/integer logical OR.	✓	✓
&	AND (logical AND)	MB – = MB – & MB – ; MB – = MB – & 1; MW – = MW – & MW – ; MW – = MW – & 00FFH;	Performs bit/integer logical AND.	✓	✓
^	XOR (logical exclusive OR)	MW – = MW – ^ MW – ; MW – = MW – ^ 00FFH;	Performs integer logical exclusive OR.	✓	✓
!	NOT (logical complement)	MB – = !MB – ; MB – = !1; MW – = !MW – ; MW – = !00FFH;	Performs bit/integer logical complement (inverts bits).	✓	✓

A.7 Data Comparison

Com-mand	Name	Programming Format	Description	Motion Programs	Sequence Programs
==	Match	IF MW – == MW – ; WHILE MW – == MW – ;	Used in an IF or WHILE conditional expression. If the left side and right side match, the condition is "true."	✓	✓
<>	Mismatch	IF MW – <> MW – ; WHILE MW – <> MW – ;	Used in an IF or WHILE conditional expression. If the left side and right side do not match, the condition is "true."	✓	✓
>	Greater Than	IF MW – > MW – ; WHILE MW – > MW – ;	Used in an IF or WHILE conditional expression. If the left side is greater than the right side, the condition is "true."	✓	✓
<	Less Than	IF MW – < MW – ; WHILE MW – < MW –	Used in an IF or WHILE conditional expression. If the left side is less than the right side, the condition is "true."	✓	✓
>=	Greater Than or Equal To	IF MW – >= MW – ; WHILE MW – >= MW – ;	Used in an IF or WHILE conditional expression. If the left side is greater than or equal to the right side, the condition is "true."	✓	✓
<=	Less Than or Equal To	IF MW – <= MW – ; WHILE MW – <= MW – ;	Used in an IF or WHILE conditional expression. If the left side is less than or equal to the right side, the condition is "true."	✓	✓

A.8 Data Operations

Com-mand	Name	Programming Format	Description	Motion Programs	Sequence Programs
SFR	Right Shift	SFR MB – N – W – ;	Shifts the bit variable in the specified num-bers to the right.	✓	✓
SFL	Left Shift	SFL MB – N – W – ;	Shifts the bit variable in the specified num-bers to the left.	✓	✓
BLK	Block Move	BLK MW – MW – W – ;	Copies the areas of specified blocks begin-ning with the specified transfer source to the specified destination.	✓	✓
CLR	Clear	CLR MW – W – ;	Clears the desired area to 0 (zero) beginning with the specified register.	✓	✓
ASCII	ASCII Convert 1	ASCII '*Character string*' MW – ;	Converts the specified characters to ASCII code, and stores to the specified register.	✓	✓

A.9 Basic Functions

Com-mand	Name	Programming Format	Description	Motion Programs	Sequence Programs
SIN	Sine	SIN (MW –); SIN (90) ;	Obtains the sine of integer or real number data. The specifications differ depending on the data type: Integer or real number.	✓	✓
COS	Cosine	COS (MW –); COS (90) ;	Obtains the cosine of integer or real number data. The specifications differ depending on the data type: Integer or real number.	✓	✓
TAN	Tangent	TAN (MF –); TAN (45.0) ;	Obtains the tangent of real number data. Only a real number type register can be designated.	✓	✓
ASN	Arc Sine	ASN (MF –); ASN (90.0) ;	Obtains the arc sine of the real number data. Only a real number type register can be designated.	✓	✓
ACS	Arc Cosine	ACS (MF –); ACS (90.0) ;	Obtains the arc cosine of real number data. Only a real number type register can be designated.	✓	✓
ATN	Arc Tangent	ATN (MW –); ATN (45) ;	Obtains the arc tangent of integer or real number data. The specifications differ depending on the data type: Integer or real number.	✓	✓
SQT	Square Root	SQT (MW –); SQT (100) ;	Obtains the root of integer or real number data. The specifications differ depending on the data type: Integer or real number.	✓	✓
BIN	BCD-to-Binary	BIN (MW –);	Converts BCD data to binary data.	✓	✓
BCD	Binary-to-BCD	BCD (MW –);	Converts binary data to BCD data.	✓	✓
S { }	Set Bit	S {MB – } = MB – & MB – ;	If the logical operation result is true, the designated bit turns ON. The designated bit does not turn OFF even if the logical operation result is false.	✓	✓
R { }	Reset Bit	R {MB – } = MB – & MB – ;	If the logical operation result is true, the designated bit turns OFF. The designated bit does not turn ON even if the logical operation result is false.	✓	✓
PON	Rising Pulse	MB – = PON (MB – MB –); or IF PON (MB – MB –) == 1; … ; IEND;	The bit output turns ON for one scan when the bit input status changes from OFF to ON.	–	✓
NON	Falling Pulse	MB – = NON (MB – MB –); or IF NON (MB – MB –) == 1; … ; IEND;	The bit output turns ON for one scan when the bit input status changes from ON to OFF.	–	✓
TON	ON-Delay Timer	MB – = MB – & TON (– MB –);	Counts the period of time when the bit input is ON. The bit output turns ON when the counted value is equal to the set value. Counting unit: 10 ms.	–	✓
TOF	OFF-Delay Timer	MB – = MB – & TOF (– MB –);	Counts the period of time when the bit input is OFF. The bit output turns OFF when the counted value is equal to the set value. Counting unit: 10 ms.	–	✓

A.10 C-Language Control Commands

Com-mand	Name	Programming Format	Description	Motion Programs	Sequence Programs
CTSK	C-Language Task Control	CTSK EXECUTE TYPE, C_NAME, COMPLETE ERROR ERR_CODE;	Controls the operations of C-language user task, such as start and stop.	✓	✓
CFUNC	C-Language Function Call	CFUNC EXECUTE OPTION1 OPTION2, C_NAME C_ARG1 C_ARG2, COMPLETE ERROR C_RETURN;	Calls the C-language user function.	✓	✓

B Sample Programs

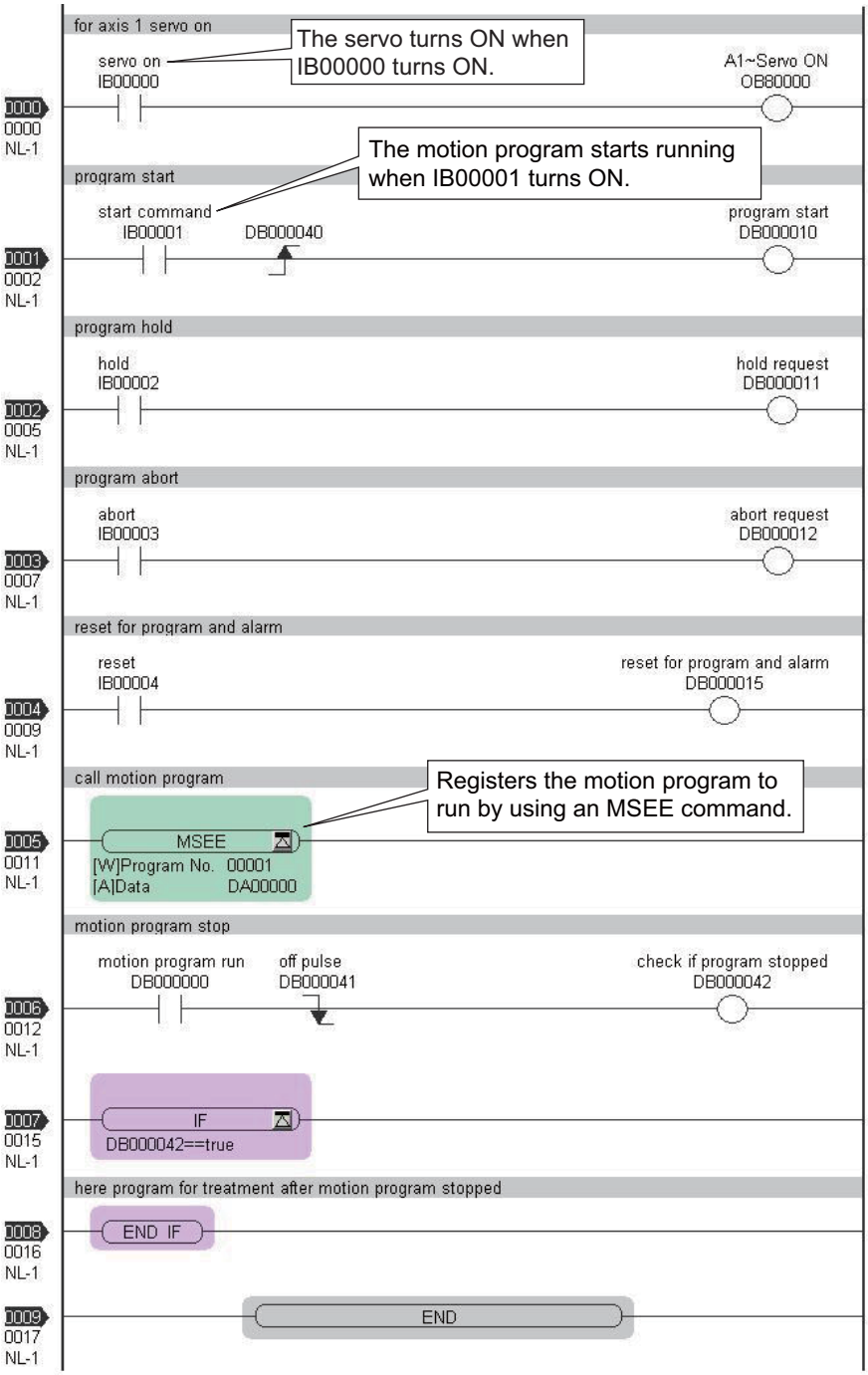
The following table shows the sample programs.

Sample Programs	Description	Reference Section
Programs to control motion programs	Sample ladder and sequence programs to control motion program.	B.1
Parallel processing	Sample motion program for parallel processing using a PFORK command.	B.2
Speed control by using a motion program	Sample motion program for speed control.	B.3
Simple synchronized operation using a virtual axis	Sample program to move two axes in synchronization using the SVR module.	B.4
Sequence programs	Sample sequence programs for JOG and STEP operations of a single axis servomotor.	B.5

B.1 Programs for Controlling Motion Program Execution

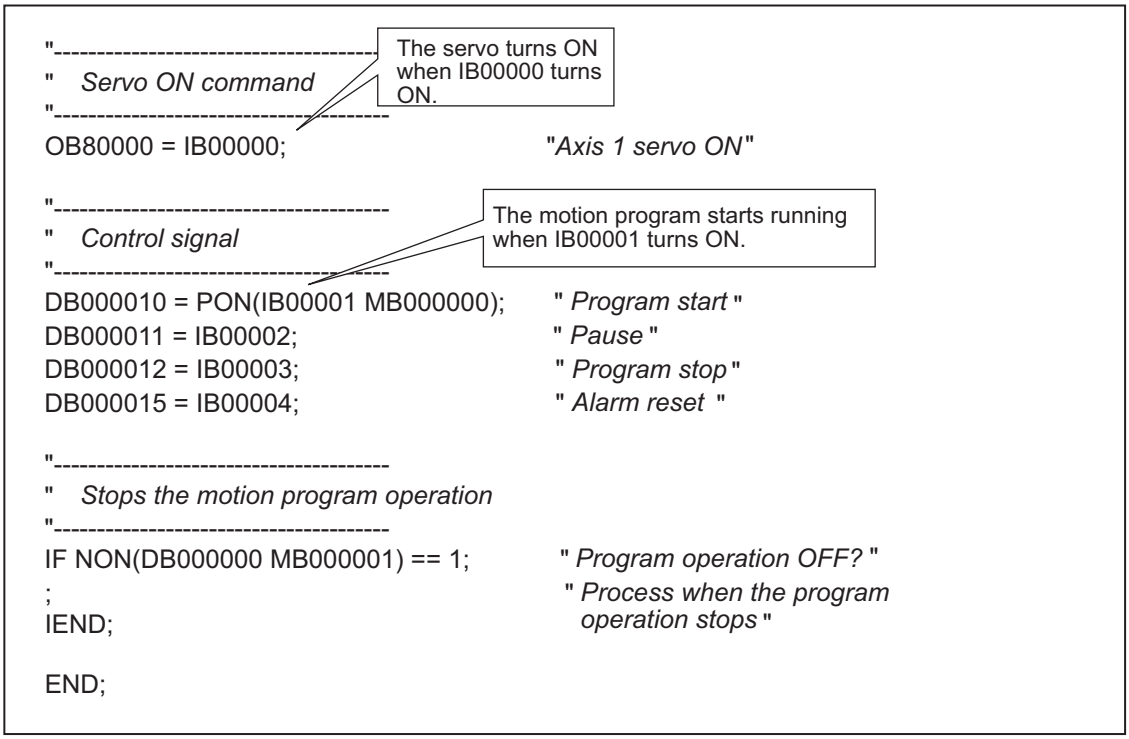
Sample ladder and sequence programs to control execution of motion program are shown below.

(1) Ladder Program



App

(2) Sequence Program



IMPORTANT

No MSEE command can be embedded in sequence programs.
In this case, embed an MSEE command in the H drawing.

B.2 Parallel Processing

A sample motion program for parallel execution accomplished by using a PFORK command is shown below.

```
    ABS;                                    " Absolute mode
    PFORK 001 002 003 004;
001:FMX T10000K;                            " Sets the max. interpolation feed speed
    PLN [A1] [B1];                          " Sets the plane for circular interpolation
    MCC [A1] 0 [B1]0 U100000 V0 F10000K;    " Circular interpolation
    JOINTO 005;
002:FMX T10000K;                            " Sets the max. interpolation feed speed
    MVS [C1] 131072 [D1] 20000 F10000K;     " Linear interpolation
    JOINTO 005;
003:IOW IL8816>130000;                      "Monitors C1 axis position
    OB00000=1;                              " When reached, OB00000 turns ON.
    JOINTO 005;
004:IOW IL8896>12000;                       "Monitors D1 axis position
    OB00001=1;                              " When reached, OB00001 turns ON.
    JOINTO 005;
005:PJOINT;
    END;
```

Starts parallel processing

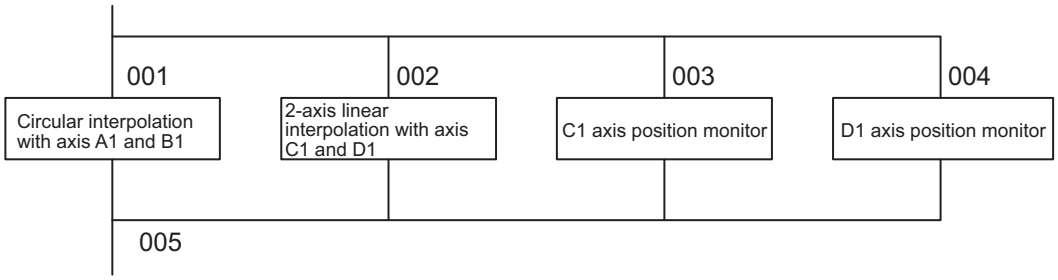
Program 1: Circular interpolation

Program 2: Linear interpolation

Program 3: C1 axis position monitor

Program 4: D1 axis position monitor

The following diagram illustrates the operations executed by the above sample program.



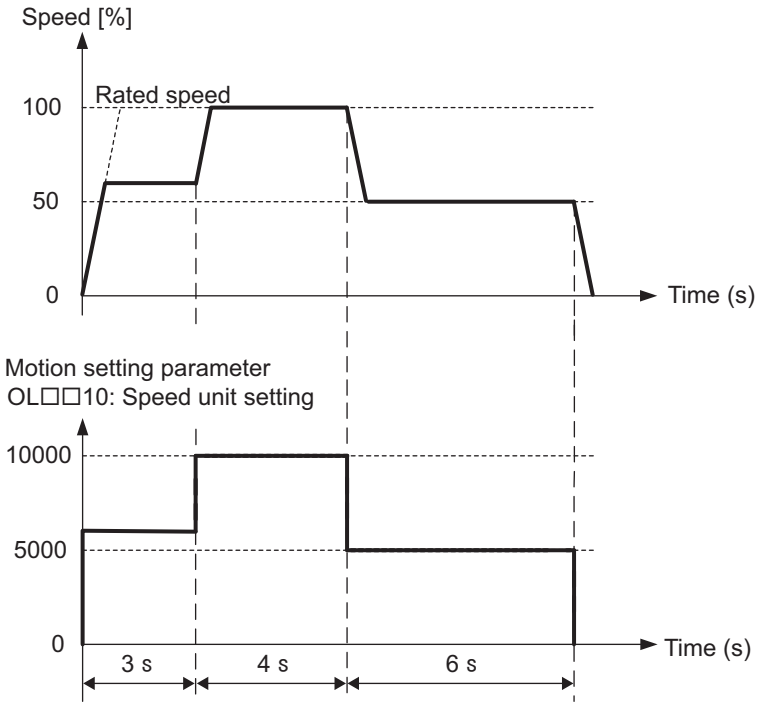
App

B.3 Motion Program for Speed Control

A sample motion program for speed control is shown below.
For this sample program, bits 0 to 3 (Speed unit selection) of motion setting parameter OW□□03 are set to 0.01% (a percentage of the rated speed).

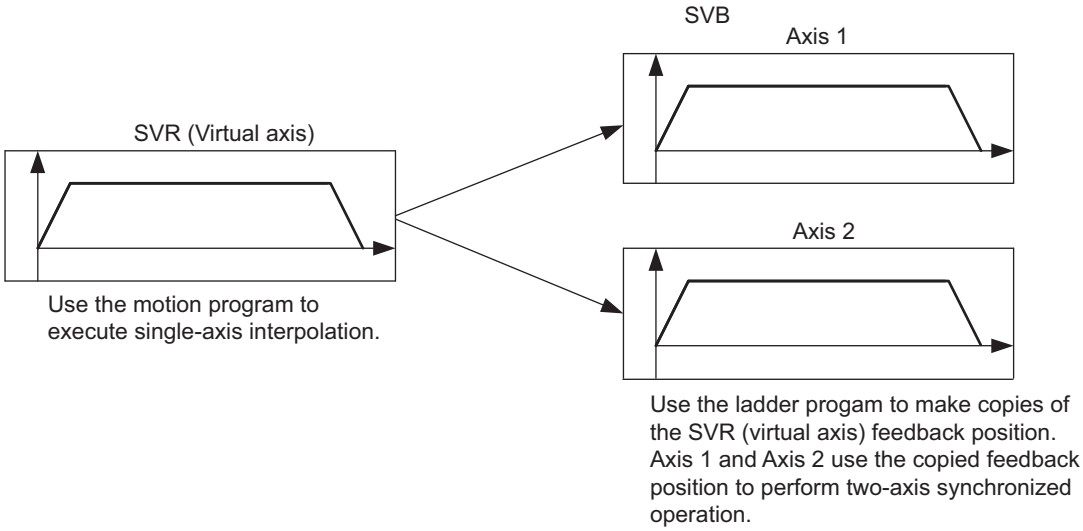
```
OW8008=23;          "Speed control mode
OL8010=6000;        "Changes the speed to 60% of rated speed
TIM T300;           "Waits for three seconds
OL8010=10000;       "Changes the speed to the rated speed
TIM T400;           " Waits for four seconds
OL8010=5000;        "Changes the speed to 50% of rated speed
TIM T600;           " Waits for six seconds
OW8008=0;           "Cancels the speed control mode
END;
```

The following diagram illustrates the moving pattern implemented by the above sample program.



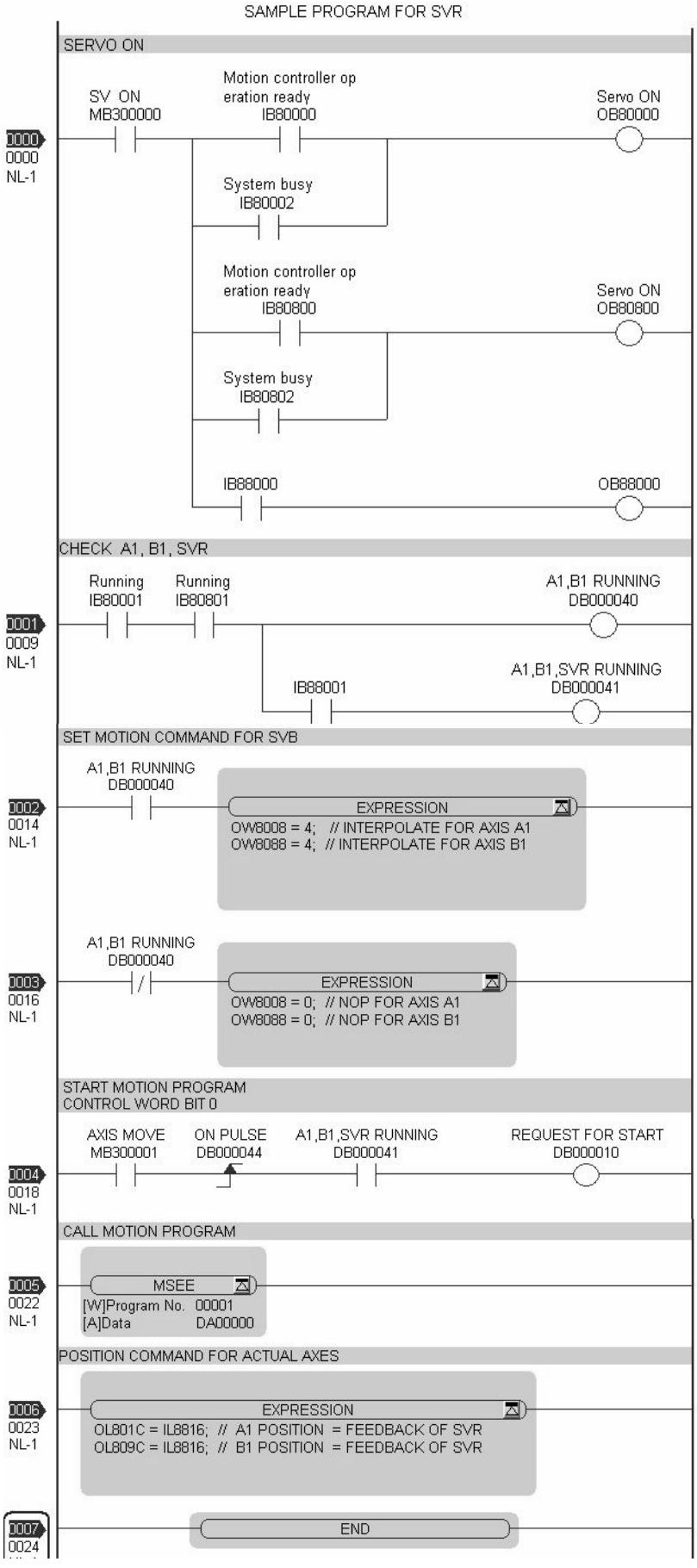
B.4 Simple Synchronized Operation Using a Virtual Axis

With this sample program, a motion program is used to move an SVR (virtual axis), and two copies of the SVR feedback position are made by a ladder program for two actual axes to perform a two-axis synchronized operation.



- Motion Program

```
FMX T10000K;                "Sets the max. interpolation feed speed. (K=1000)
INC;                        "Incremental mode
IAC T500;                   "Interpolation acceleration time = 500 ms
IDC T500;                   "Interpolation deceleration time = 500 ms
MVS [SVR] 1000K F10000K;    "Interpolation with movement amount 1000000
END;
```


SAMPLE PROGRAM FOR SVR
SERVO ON
SV ON
MB300000
Motion controller op
eration ready
IB80000
Servo ON
OB80000
0000
0000
NL-1
System busy
IB80002
Motion controller op
eration ready
IB80800
Servo ON
OB80800
System busy
IB80802
IB88000
OB88000
CHECK A1, B1, SVR
Running
IB80001
Running
IB80801
A1,B1 RUNNING
DB000040
0001
0009
NL-1
A1,B1,SVR RUNNING
DB000041
IB88001
SET MOTION COMMAND FOR SVB
A1,B1 RUNNING
DB000040
0002
0014
NL-1
EXPRESSION
OW8008 = 4; // INTERPOLATE FOR AXIS A1
OW8088 = 4; // INTERPOLATE FOR AXIS B1
A1,B1 RUNNING
DB000040
0003
0016
NL-1
EXPRESSION
OW8008 = 0; // NOP FOR AXIS A1
OW8088 = 0; // NOP FOR AXIS B1
START MOTION PROGRAM
CONTROL WORD BIT 0
AXIS MOVE
MB300001
ON PULSE
DB000044
A1,B1,SVR RUNNING
DB000041
REQUEST FOR START
DB000010
0004
0018
NL-1
CALL MOTION PROGRAM
0005
0022
NL-1
MSEE
[W]Program No. 00001
[A]Data DA00000
POSITION COMMAND FOR ACTUAL AXES
0006
0023
NL-1
EXPRESSION
OL801C = IL8816; // A1 POSITION = FEEDBACK OF SVR
OL809C = IL8816; // B1 POSITION = FEEDBACK OF SVR
0007
0024
END

B.5 Sequence Programs

In this sample program, sequence programs are used for JOG and STEP operation of servomotor with single axis.

Sequence main program (SPM001)

```
"SPM001: Main program"

SSEE SPS002;              "Settings common to all axes
SSEE SPS003;              "JOG & STEP operation process
END;
```

Sequence subprogram (SPS002)

```
"SPS002: Settings common to all axes"

"---------------------------------------
"  Motion command 0 (No command) detection
"---------------------------------------
IF IW8008 == 0;
        MB300010 = 1;
ELSE;
        MB300010 = 0;
IEND;
                                  The Servo turns ON when
                                  MB300000 turns ON.
"---------------------------------------
"  Servo ON command
"---------------------------------------
OB80000 = MB300000 & (IB80000 | IB80002);        "Servo ON

"---------------------------------------
"  Alarm reset
"---------------------------------------
OB8000F = MB300001;                              "Alarm reset

"-------------------------------------------------------------------------
"  Speed unit & Acceleration/deceleration unit selection
"
"    Bit 0 to 3: Speed unit selection (0: reference unit/s, 1: Reference unit/min., 2: Designated in %)
"    Bit 4 to 7: Acceleration/deceleration unit selection (0: Reference unit /s2, 1: ms)
"-------------------------------------------------------------------------
DW00010 = OW8003 & FF00H;                        "Function setting 1 work
OW8003 = DW00010 | 0011H;                        "Function setting 1

"---------------------------------------
"  Linear acceleration/deceleration setting
"---------------------------------------
IF MB300020 == 1;
      OL8036 = 100;                              "Linear acceleration rate/Acceleration time constant
      OL8038 = 100;                              "Linear deceleration rate/Deceleration time constant
IEND;

RET;
```

App

Sequence subprogram (SPS003)

```
"SPS003: JOG & STEP operation process "

"----------------------------------------
"  JOG operation
"----------------------------------------
IF IB80001 & ( (DB000010 & !DB000011) | (!DB000010 & DB000011) ) == 1;
      DB000000 = 1;
ELSE;
      DB000000 = 0;
IEND;

DB000001 = PON( DB000000 DB000050 ) & MB300010; "JOG operation starts
DB000002 = NON( DB000000 DB000051 );            "JOG operation stops

IF DB000001 == 1;
      OL8010 = 1000;
      OW8008 = 7;                               "Motion command FEED
IEND;
IF DB000002 == 1;
      OW8008 = 0;                               "Motion command NOP
IEND;

"----------------------------------------
"  STEP operation
"----------------------------------------
IF IB80001 & ( (DB000012 & !DB000013) | (!DB000012 & DB000013) ) == 1;
      DB000008 = 1;
ELSE;
      DB000008 = 0;
IEND;

DB000009 = PON( DB000008 DB000058 ) & MB300010; "STEP operation starts
DB00000A = NON( DB000008 DB000059 );            "STEP operation stops

IF DB000009 == 1;
      OL8010 = 1000;                            "STEP speed setting
      OW8044 = 1000;                            "STEP moving amount setting (1000 pulses)
      OW8008 = 8;                               "Motion command STEP
IEND;

IF DB00000A == 1;
      OW8008 = 0;                               "Motion command NOP
IEND;

"----------------------------------------
"   Reverse rotation selection
"----------------------------------------
OB80092 = ( DB000000 & DB000011 ) | ( DB000008 & DB000013 );"Reverse rotation selection

RET;
```

Starts JOG operation (forward rotation) when DB000010 turns ON.

Starts JOG operation (reverse rotation) when DB000011 turns ON.

Starts STEP operation (forward rotation) when DB000012 turns ON.

Starts STEP operation (reverse rotation) when DB000013 turns ON.

C Differences between MP900 Series and MP2000 Series Machine Controllers

This section describes motion program differences between the MP900 series and the MP2000 series Machine Controllers.

C.1 Motion Programs

Item	MP900 Series	MP2000 Series	Remarks
Number of tasks	No limitation	16 tasks	Number of simultaneously executable programs
Group definition Max. number of axes per group	MP910: 28 axes MP920: 48 axes MP930: 14 axes MP940: 1 axis	16 axes	–
Work size of MSEE command in ladder programs	2 words	4 words	See *4.3.3 Work Registers*.
Interpolation override	Always enabled	Enabled or disabled can be selected.	See *4.3.3 Work Registers*.
Register for interpolation override value designation	Designated in the **Group Definition** dialog box.	Designated using the 3rd word of an MSEE work register.	See *4.3.3 Work Registers*.
Program execution registration function (M-EXECUTOR Module)	Not supported	Supported	Can be used with MP2100, MP2100M, MP2300S, MP2310, and MP2400 only.
Nesting of PFORK command (PFORK execution during parallel processing)	Permitted	Prohibited	–
Axis move command in the two parallel subprograms	Prohibited	Permitted	–
Subprogram call (MSEE) from the two parallel subprogram	Permitted	Prohibited	–
Numbers below decimal point when real number data is stored in an integer register	Rounded-off	Cut-down (Truncate) or rounded-off (Rounding)	The default for the MP2000 series is cut-down.

C.2 Sequence Programs

Item	MP900 Series	MP2000 Series	Remarks
Applicable/Not applicable	Not applicable	Applicable	Can be used with MP2100, MP2100M, MP2300S, MP2310, and MP2400 only.

C.3 Motion Programming Commands

Item	MP900 Series	MP2000 Series	Remarks
ACC, DCC	• Executable with SVA-01, SVA-02, and SVB-01 Modules. • Unexecutable with PO-01 Module.	Executable	–
SCC	• Executable with SVB-01. • Unexecutable with SVA-01, SVA-02, and PO-01.	Executable	–
Speed unit for VEL command	10^n reference units/min.	Can be selected from: • 10^n reference units/min. • Reference units/s • 0.01% • 0.0001%	• Depends on the difference in functions between MP900 series Motion Modules and MP2000 series Motion Modules. • See *8.1.6 Set Velocity (VEL)*.
Speed unit for VEL command when the reference unit is set to pulse (10^n reference units/min.)	MP920 PO-01 Module: 100 pulses/min. Motion modules other than MP920 PO-01: 1000 pulses/min.	1000 pulses/min.	• Depends on the difference in functions between MP900 series Motion Modules and MP2000 series Motion Modules. • See *8.1.6 Set Velocity (VEL)*.
Acceleration/Deceleration rate designation method for ACC and DCC commands	MP930 SVB module: Acceleration/deceleration time (ms) calculated on the base of the feed speed Modules other than MP930 SVB Module: Acceleration/deceleration time (ms) calculated on the base of the rated speed	Can be selected from followings: • Acceleration/deceleration time (ms) calculated on the base of the rated speed • Acceleration/deceleration rate (reference unit/s^2)	• Depends on the difference in functions between MP900 series Motion Modules and MP2000 series Motion Modules. • See *8.1.3 Acceleration Time Change (ACC)* and *8.1.4 Deceleration Time Change (DCC)*.
Unit of moving amount specified for external positioning command EXM	In unit of pulse	In reference unit	Depends on the difference in functions between MP900 series Motion Modules and MP2000 series Motion Modules.

C.4 Group Definitions

Item	MP900 Series	MP2000 Series	Remarks
Group definition saving operation while online	Prohibited	Permitted	–
Automatic generation of ladder program	Possible (Can be specified in the **Group Definition** dialog box.)	Not possible	–

C.5 Debug Function

Item	MP900 Series	MP2000 Series	Remarks
Single-block operation applicable commands	Axis move commands only	All commands	–
Designation to ignore single-block signal	By coding an SNG command for each block that ignores the single-block signal	By executing SNGD and SNDE commands in the enclosed blocks.	See *8.4.14 Single-block Signal Disabled (SNGD)/Single-block Signal Enabled (SNGE)*.
Number of settable break points	1	4	–

C.6 Motion Program Alarms

Item	MP900 Series	MP2000 Series	Remarks
Motion program alarm storage	Storage designated in the **Group Definition** dialog box	S register	See *10.2.3 Confirming the Alarm Code*.

D Precautions

D.1 General Precautions

(1) Saving the Changes in Programs to the Flash Memory

After changing an application such as a motion program or sequence program, always save the changed application to the flash memory. If the changed application is not saved to the flash memory and the power to the Machine Controller is turned OFF, the changed application will be lost.

(2) Prohibited Use of Debug Function for a System in Operation

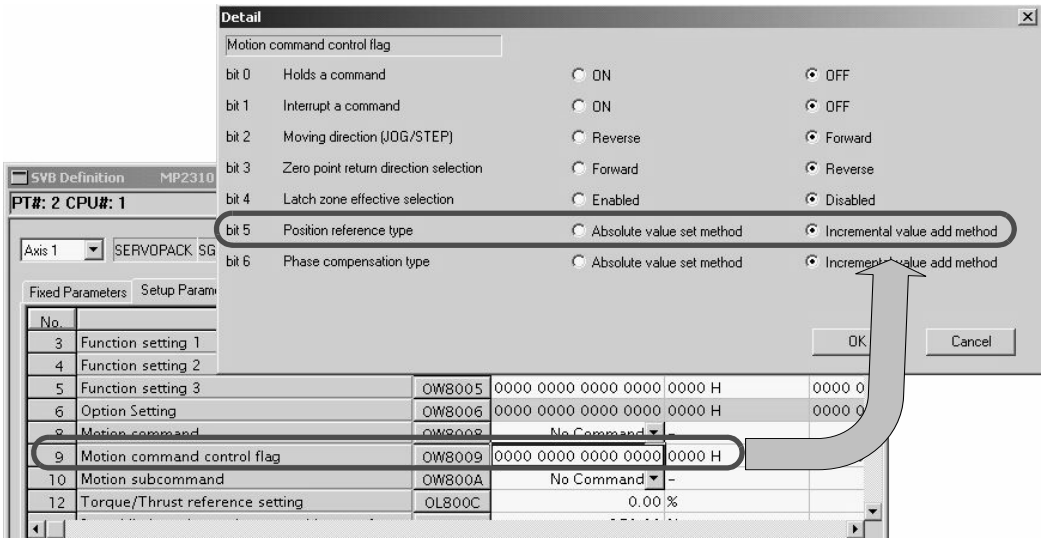
Never use the debug function for a system in operation. Debugging will cause changes in program operation, such as in command execution timing, resulting in malfunction or failure of the system.

For debugging, use the exclusive system for debugging.

D.2 Precautions on Motion Parameter Settings

(1) Set bit 5 (Position reference type) of motion setting parameter OW□□09 to Incremental value add method

When using a motion program, always set bit 5 (Position reference type) of motion setting parameter OW□□09 to Incremental value add method. Motion programs manage position information using the incremental value add method. Therefore, operation using a motion program cannot be guaranteed if bit 5 of motion setting parameter OW□□09 is set to Absolute value set method.



With a motion program, an ABS or INC command is used to set the absolute mode or incremental mode:
Code an ABS command to set the absolute mode (ABS).
Code an INC command to set the incremental mode (INC).

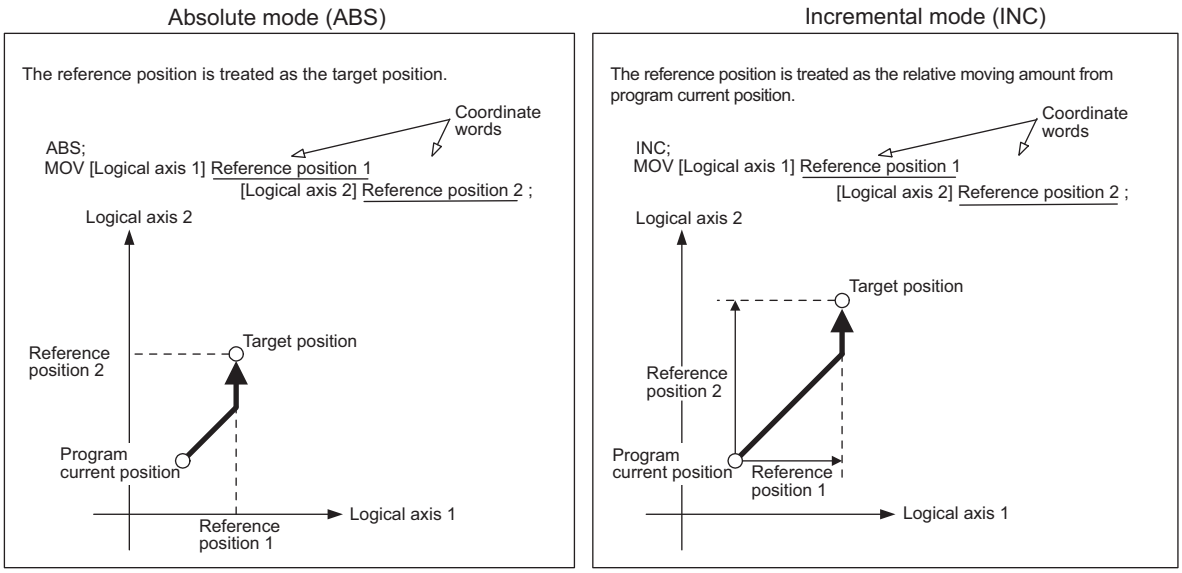
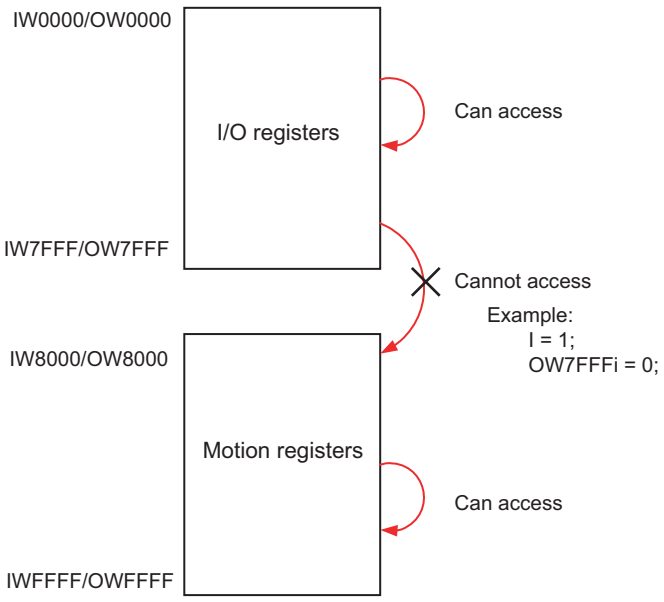


Fig. D.1 Movement Modes for Axis Move Commands

(2) Do not access a motion register from the I/O register by using a subscript

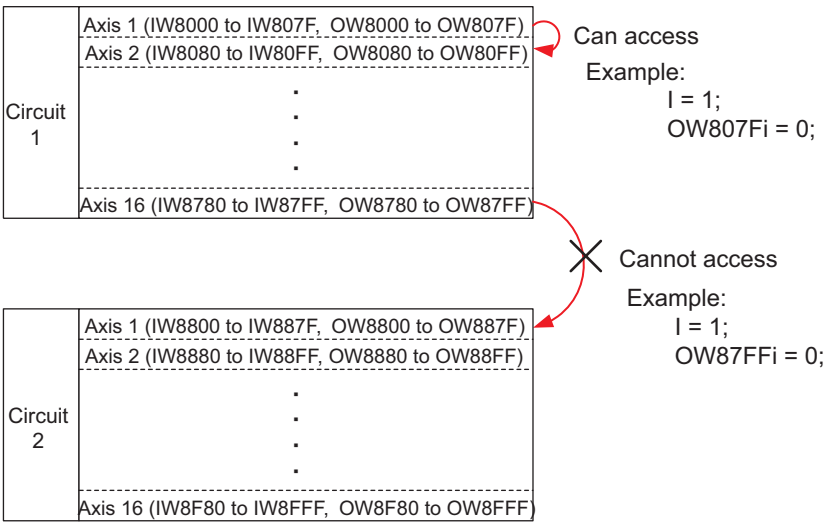
I/O registers and motion registers are not assigned to consecutive memory locations.
Therefore, when using a subscript, access is limited within the respective register range of I/O register and motion register.



(3) Do not access a motion register of a different circuit by using a subscript

As with the relationship between I/O registers and motion registers, motion registers of different circuits are not assigned to consecutive memory locations.
Therefore, when using a subscript, access is limited within the same motion register range of each circuit.
If the circuit numbers of two registers are the same, it is possible to access a register of a different axis.

Circuit No.	Axis 1	Axis 2	···	Axis 16
1	OW8000 to OW807F	OW8080 to OW80FF	···	OW8780 to OW87FF
2	OW8800 to OW887F	OW8880 to OW88FF	···	OW8F80 to OW8FFF
3	OW9000 to OW907F	OW9080 to OW90FF	···	OW9780 to OW97FF
4	OW9800 to OW987F	OW9880 to OW98FF	···	OW9F80 to OW9FFF
5	OWA000 to OWA07F	OWA080 to OWA0FF	···	OWA780 to OWA7FF
6	OWA800 to OWA87F	OWA880 to OWA8FF	···	OWAF80 to OWAFFF
7	OWB000 to OWB07F	OWB080 to OWB0FF	···	OWB780 to OWB7FF
8	OWB800 to OWB87F	OWB880 to OWB8FF	···	OWBF80 to OWBFFF
9	OWC000 to OWC07F	OWC080 to OWC0FF	···	OWC780 to OWC7FF
10	OWC800 to OWC87F	OWC880 to OWC8FF	···	OWCF80 to OWCFFF
11	OWD000 to OWD07F	OWD080 to OWD0FF	···	OWD780 to OWD7FF
12	OWD800 to OWD87F	OWD880 to OWD8FF	···	OWDF80 to OWDFFF
13	OWE000 to OWE07F	OWE080 to OWE0FF	···	OWE780 to OWE7FF
14	OWE800 to OWE87F	OWE880 to OWE8FF	···	OWEF80 to OWEFFF
15	OWF000 to OWF07F	OWF080 to OWF0FF	···	OWF780 to OWF7FF
16	OWF800 to OWF87F	OWF880 to OWF8FF	···	OWFF80 to OWFFFF



(4) Do not change the motion setting parameter OL□□1C “Position reference setting” while axis motion is in progress in a motion program.

If OL□□1C “Position reference setting” is changed in another program (e.g. a ladder program) while axis motion is in progress in a motion program, the axes will move in accordance with the change. This will result in a discrepancy between the actual position and the position specified in the motion program.

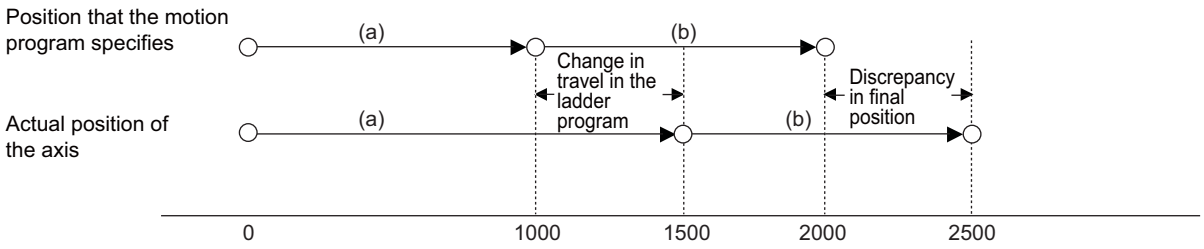
Example:

If the travel distance of the A1 axis specified by OL□□1C “Position reference setting” in the motion program below is changed from +1000 to +1500 while executing (a) in the motion program, the A1 axis moves to the position +1500. At this point a discrepancy arises in relation to the reference position in the motion program (+1000). Then (b) in the motion program will be executed. As a result, the actual final position of the A1 axis will be at a different position from that specified in the motion program.

```
INC;
ZRN [A1]0;

MOV [A1]1000; . . . (a)
MOV [A1]1000; . . . (b)

END;
```



Index

Symbols

! ---------- 2-6, 8-127
register- ---------- 2-4, 6-2, 8-104
& ---------- 2-6, 8-125
* ---------- 2-6, 8-120
+ ---------- 2-6, 8-118
- (subtract) ---------- 2-6, 8-119
/- ---------- 2-6, 8-121
< ---------- 2-6, 8-128
<= ---------- 2-6, 8-128
<> ---------- 2-6, 8-128
= ---------- 2-6, 8-117
== ---------- 2-6, 8-128
> ---------- 2-6, 8-128
>= ---------- 2-6, 8-128
^ ---------- 2-6, 8-126
|- ---------- 2-6, 8-123

Numerics

0.0001% of rated speed - ---------- 7-12
0.01% of rated speed- ---------- 7-12
10n reference unit/min. ---------- 7-12

A

ABS- ---------- 2-6, 8-3
ABS mode ---------- 8-3
absolute mode (ABS) ---------- 8-3
ACC ---------- 2-6, 8-11
Acceleration Time Change (ACC) ---------- 8-11
acceleration time/deceleration time ---------- 8-30
acceleration/deceleration degree unit selection ---------- 7-12
acceleration/deceleration setting ---------- 7-12
acceleration/deceleration type ---------- 8-47, 8-53
ACS- ---------- 2-6, 8-141
add (+) ---------- 8-118
Add Register ---------- 9-21
alarm code ---------- 10-12
alarm code list for motion program ---------- 10-15
alarm display ---------- 9-35
alarm reset request ---------- 4-8
allocated interlock contact ---------- 4-12
Allocation DISABLE ---------- 9-14
Allocation register ---------- 4-12
AND (&) ---------- 8-125
APOS- ---------- 8-82
Arc Cosine (ACS) ---------- 8-141
Arc Sine (ASN) ---------- 8-140
Arc Tangent (ATN)- ---------- 8-142
arithmetic operations ---------- 8-117
ASCII- ---------- 2-6, 8-134
ASCII code ---------- 8-134
ASCII Conversion 1 (ASCII) ---------- 8-134
ASN- ---------- 2-6, 8-140
ATN ---------- 2-6, 8-142
axis alarm monitor ---------- 9-31, 9-33
axis control commands ---------- 8-78
axis move commands ---------- 8-46
Axis No. ---------- 7-14
axis operation monitor ---------- 9-31
Axis Operation Monitor function ---------- 1-7
Axis Operation Monitor/Alarm Monitor function ---------- 2-5
axis setting commands ---------- 8-3
axis status- ---------- 9-33
axis type selection ---------- 7-9, 8-5

B

basic flow of troubleshooting- ---------- 10-2
basic functions ---------- 8-136
Batch transfer ---------- 3-11
BCD ---------- 2-6, 8-136, 8-146
BCD data ---------- 8-145, 8-146
BCD to Binary (BIN) ---------- 8-145
BIN ---------- 2-6, 8-145
binary data ---------- 8-145, 8-146
binary data conversion ---------- 8-136
Binary to BCD (BCD) ---------- 8-146
bit ---------- 6-3
Bit Left Shift (SFL) ---------- 8-131
Bit Right Shift (SFR) ---------- 8-130
BLK ---------- 2-6, 8-132
block ---------- 7-2, 9-4
block format ---------- 7-2
Block Move (BLK) ---------- 8-132
block number ---------- 10-12
branching commands (IF ELSE IEND) ---------- 8-87
Break- ---------- 9-19
break point setting ---------- 1-7
built-in SVB ---------- 1-8, 2-2

C

C registers ---------- 2-4, 6-13
C variables (C registers) ---------- 6-13
C-Language Function Call (CFUNC) ---------- 8-160
C-Language Task Control (CTSK) ---------- 8-158
call ---------- 1-8
center point position- ---------- 8-56
center position designation ---------- 8-55, 8-65
CFUNC ---------- 2-6, 8-160
character
  D ---------- 7-5
  F ---------- 7-5
  MPS ---------- 7-6
  N ---------- 7-5
  P ---------- 7-5
  R- ---------- 7-5
  SPS ---------- 7-6
  SS ---------- 7-5
  T- ---------- 7-5
  U ---------- 7-5
  V ---------- 7-5
  W ---------- 7-5
child drawing ---------- 1-9
Circuit ---------- 7-13
circular interpolation ---------- 1-13
Clear (CLR) ---------- 8-133
Clockwise/Counterclockwise Circular Interpolation with Center Position Designation (MCW, MCC) ---------- 8-55
Clockwise/Counterclockwise Circular Interpolation with Radius Designation (MCW, MCC) ---------- 8-61
Clockwise/Counterclockwise Helical Interpolation with Center Position Designation (MCW, MCC) ---------- 8-65
Clockwise/Counterclockwise Helical Interpolation with Radius Designation (MCW, MCC) ---------- 8-68
CLR ---------- 2-6, 8-133
command format ---------- 9-8
command input assistant ---------- 9-6

Command Input Assistant function - - - - - - - - - - - - - - - - - - 1-7, 1-16
command input assistant function - 2-5
command types - 7-17
comment check box and comment input field - - - - - - - - - - - - - - 9-10
comments - 7-2, 7-6
communication settings - 3-3
compiling - 3-6
composite moving amount - 8-51
configuration of motion program alarms - - - - - - - - - - - - - - - - - 10-15
confirming the alarm code - 10-9
constant registers - 6-2, 8-104
constants - 7-7
control signal - 4-8, 9-14, 10-12
Coordinate Plane Setting (PLN) - 8-86
coordinate words - 7-2, 7-4
COS - 2-6, 8-138
Cosine (COS) - 8-138
creating a motion program - 3-6
creating project files - 3-4
CTSK - 2-6, 8-158
Current Position Set (POS) - 8-78

D

D registers - 2-4, 6-2, 6-14, 8-104
D variables (D registers) - 6-14
data comparison commands - 8-128
data operations - 8-130
data registers - 6-2, 8-104
data types - 6-3, 8-102
data variables (M registers) - 6-8
DCC - 2-6, 8-17
Debug function - 1-7, 1-16, 2-5, 9-15
debug mode - 3-12, 9-17
debugging the program - 3-12
Deceleration Time Change (DCC) - 8-17
decimal integers - 7-7
DEFAULT - 8-95
deg - 7-9
degree - 8-136
direct designation - 5-4, 9-13
divide (/) - 8-121
double integer - 6-3
Drive control panel - 2-5, 9-24
Drive control panel function - 1-7
Dwell Time (TIM) - 8-112

E

easy programming functions - 1-7, 1-16
electronic gear - 7-10, 8-33
encoder cable - 3-3
END - 2-6, 8-110
end of block - 7-2, 7-6
end position - 8-56
EOX - 2-6, 8-115
error code - 4-15
Error List dialog box - 3-6
Execute - 9-19
executing block number - 4-15
executing main program No. - 10-12
executing program line - 9-16
executing program number - 4-14, 4-15
executing the programs - 3-14
execution method - 2-3, 2-4
execution registry screen - 1-8
execution scans - 7-17
Execution type - 9-13
EXM - 2-6, 8-76
external function register - 8-104
External Positioning (EXM) - 8-76
external positioning signal - 8-77

F

F designation - 8-37, 8-52
Falling Pulse (NON) - 8-151
filter time constant - 8-24
filter type selection - 8-28
finite length axis - 7-9, 8-5
FMX - 2-6, 8-35
Forced end - 9-19
Fork number - 9-20
FUNC - 2-6, 8-109
function input register - 8-104
function keys - 9-16
function output register - 8-104
Function selection flag 1 - 8-5

G

global variables - 6-4, 7-7
grandchild drawing - 1-9
group definitions - 1-11, 3-5, 7-13
Group Name - 7-13
group selection - 9-4
groups - 1-11

H

H drawing - 1-9
handling system - 1-12
Help button - 9-11
hexadecimal integers - 7-7
high-speed processing - 2-3
high-speed processing drawing - 1-9
how to directly change the acceleration time setting - - - - - - - - - 8-16
how to directly change the deceleration time setting - - - - - - - - - 8-22

I

I registers - 2-4, 6-9
I/O services - 1-10
I/O Variable Wait (IOW) - 8-113
IAC - 2-6, 8-40
IDC - 2-6, 8-43
IF ELSE IEND - 2-6, 8-87
IFP - 2-6, 8-37
in debug mode - 4-7, 5-5
in-position check - 8-48, 8-50
In-Position Check (PFN) - 8-82
in-position check width - 8-84
INC - 2-6, 8-7
inch - 7-9
incremental mode (INC) - 8-3, 8-7
indirect designation - 4-11, 5-4, 9-13
indirect designation of a program number using a register - - - - - 4-11
Individual transfer - 3-11
infinite length axis - 7-9, 8-5
Infinite Length Axis Reset Position - 7-9
infinite length axis reset position (POSMAX) - - - - - - - - - - - - - - 8-5
INP - 2-6, 8-84
input data - 6-9
input registers - 6-2, 8-104
input variables (I registers) - 6-9
installation - 3-2
installing MPE720 Version 6 - 3-3

integer - - -

MPE720 Ver.6 Lite ---------- 2-5
MPOS ---------- 8-82
ms ---------- 7-12
MSEE ---------- 2-6, 8-99
MSEE call stack ---------- 9-20
MSEE command ---------- 1-2
MSEE work register ---------- 1-9
multiple group operation ---------- 1-11, 4-2
multiply (*) ---------- 8-120
MVM ---------- 2-6, 8-80
MVS ---------- 2-6, 8-50
MVT ---------- 2-6, 8-74

N

NEAR signal output width ---------- 8-82, 8-84
Nest number ---------- 9-20
new project ---------- 3-2
no system work error ---------- 4-7
NON ---------- 2-6, 8-151
Normal run mode ---------- 9-17
NOT (!) ---------- 8-127
number of axes to be controlled ---------- 7-13, 9-9
number of blocks ---------- 8-133
number of digits below decimal point ---------- 7-9, 8-30
number of groups ---------- 2-3, 7-13
number of parallel processes ---------- 2-3, 2-4
number of programs ---------- 2-3
number of simultaneously controlled axes ---------- 2-3
number of tasks ---------- 2-3, 2-4
number of turns ---------- 8-58

O

O registers ---------- 2-4, 6-11
OFF-Delay Timer (TOF)
counting unit 0.01 second ---------- 8-156
ON-Delay Timer (TON)
counting unit 0.01 second ---------- 8-154
One Scan Wait (EOX) ---------- 8-115
online editing ---------- 1-6
operation mode ---------- 2-3
OR (|) ---------- 8-123
organizaing the axes into individual groups ---------- 3-5
organizing the axes into individual groups ---------- 3-5
output data ---------- 6-11
output registers ---------- 6-2, 8-104
output variables (O registers) ---------- 6-11
override ---------- 2-3, 8-32
override setting for interpolation ---------- 4-8

P

palletizing ---------- 1-12
panel processing machine ---------- 1-13
Parallel Execution (PFORK, JOINTO, PJOINT) ---------- 8-92
parent drawing ---------- 1-9
pausing/starting monitoring ---------- 9-33, 9-35
PFN ---------- 2-6, 8-82
PFORK ---------- 8-92
PFORK, JOINTO, PJOINT ---------- 2-6
PJOINT ---------- 8-92
PLD ---------- 2-6, 8-81
PLN ---------- 2-6, 8-86
PO-01 ---------- 1-8, 2-2
PON ---------- 2-6
POS ---------- 2-6
position reference value ---------- 8-3
positioning (MOV) ---------- 8-46
positioning related commands ---------- 8-29
positioning speed ---------- 8-13, 8-19, 8-29
POSMAX ---------- 8-5
PP cable ---------- 3-3
precautions for variable operations ---------- 6-6
Priority Levels of Operations ---------- 7-15
problem when starting a motion program ---------- 10-4
procedure to create the user function ---------- 8-106
program alarm is occurring ---------- 4-7, 5-5
program capacity ---------- 2-3
program continuous operation start request ---------- 4-8
program control commands ---------- 8-87
program control signal ---------- 4-15
program current position ---------- 8-3, 8-81, 10-12
Program Current Position Update (PLD) ---------- 8-81
program designation method ---------- 9-13
program development flow ---------- 3-2
program editing window ---------- 9-4
program end (END) ---------- 8-110
program execution registration function ---------- 1-7, 2-5, 9-12
program execution registry number ---------- 4-13, 9-13, 9-26
program information used by work n ---------- 4-15
program number ---------- 9-13, 9-26, 10-12
program numbers of motion programs ---------- 4-2
program numbers of sequence programs ---------- 5-2
program operation start request ---------- 4-8
program pause request ---------- 4-8
program paused ---------- 4-7
Program Property window ---------- 6-5
program running ---------- 4-7, 5-5
program single block mode selection ---------- 4-8
program single block start request ---------- 4-8
program status ---------- 4-15
program stop request ---------- 4-8
program stopped by stop request ---------- 4-7
program type ---------- 4-7, 5-5
programs created for each group ---------- 1-12
pulse ---------- 7-9

R

R{ } ---------- 2-6, 8-148
radius ---------- 8-62
radius designation ---------- 8-61, 8-68
rated speed ---------- 8-12, 8-18
real number ---------- 6-3, 7-7
reference position ---------- 8-3
reference range ---------- 2-3
reference unit ---------- 2-3, 7-9
reference unit selection ---------- 8-30
reference unit/s ---------- 7-12
reference unit/s2 ---------- 7-12
register allocation ---------- 9-14
register list ---------- 3-2
register start number of the motion monitor parameter ---------- 6-10
register start number of the motion setting parameter ---------- 6-12
register types used within user functions ---------- 8-102
registering MPM001 in the M-EXECUTOR ---------- 4-5
registering the program execution ---------- 3-7
registering to the M-EXECUTOR program execution definition ---------- 4-4
relationship between I/O registers and internal function registers ---------- 8-103
relative movement amount ---------- 8-7
remainder (MOD) ---------- 8-122
Repeat (WHILE WEND) ---------- 8-89
Reset Bit (R{ }) ---------- 8-148

RET - - -

tool icons - - -

Revision History

The revision dates and numbers of the revised manuals are given on the bottom of the back cover.

MANUAL NO. SIEP C880700 38A

Published in Japan July 2009 08-9 ◇0 -1

- July 2009: Date of publication
- 08-9: Date of original publication
- ◇0: Revision number
- -1: WEB revision number

Date of Publication	Rev. No.	WEB Rev. No.	Section	Revised Contents
July 2012	◇1	5	8.2.3 (3)	Revision: Figure and INFO on interpolation feed speed
			Back cover	Revision: Address
November 2011		4	6.2.2 (3), 6.2.6 (3)	Addition: VEL command in the colum of important
June 2011		3	8.2.4	Revision: Descriptions on setting items for MCW and MCC commands.
April 2011		2	7.1.2 (5)	Revision: Application example “TIM TML00000” → “TIM TMW00000”
November 2010		1	Front cover	Revision: Format
			8.4.11 (2)	Revision: Usable data
			Back cover	Revision: Address, format
September 2010		0	–	Printed version of the user’s manual, SIEPC88070038<0>-2, available on the web.
			All chapters	Based on Japanese user’s manual, SIJPC88070038B<2>, printed in December 2009.
			Preface	Addition: Warranty
			D.2 (4)	Addition: Precautions on the motion parameter
			Back cover	Revision: Address
September 2009	◇0	2	Preface	Addition: Warranty
			Back cover	Revision: Address
July 2009		1	Appendices A.4	Revision: Descriptions in columns of sequence program for Branching command and Repeat command
			Back cover	Revision: Address
September 2008	–	–	–	First edition

Machine Controller MP2000 Series

USER'S MANUAL

for Motion Programming

IRUMA BUSINESS CENTER (SOLUTION CENTER)
480, Kamifujisawa, Iruma, Saitama 358-8555, Japan
Phone 81-4-2962-5151 Fax 81-4-2962-6138

YASKAWA AMERICA, INC.
2121 Norman Drive South, Waukegan, IL 60085, U.S.A.
Phone (800) YASKAWA (800-927-5292) or 1-847-887-7000 Fax 1-847-887-7310

YASKAWA ELÉTRICO DO BRASIL LTDA.
Avenida Fagundes Filho, 620 São Paulo-SP CEP 04304-000, Brasil
Phone 55-11-3585-1100 Fax 55-11-5581-8795

YASKAWA EUROPE GmbH
Hauptstraβe 185, Eschborn 65760, Germany
Phone 49-6196-569-300 Fax 49-6196-569-398

YASKAWA ELECTRIC UK LTD.
1 Hunt Hill Orchardton Woods Cumbernauld, G68 9LF, United Kingdom
Phone 44-1236-735000 Fax 44-1236-458182

YASKAWA ELECTRIC KOREA CORPORATION
9F, Kyobo Securities Bldg. 26-4, Yeouido-dong, Yeongdeungpo-gu, Seoul, 150-737, Korea
Phone 82-2-784-7844 Fax 82-2-784-8495

YASKAWA ELECTRIC (SINGAPORE) PTE. LTD.
151 Lorong Chuan, #04-01, New Tech Park 556741, Singapore
Phone 65-6282-3003 Fax 65-6289-3003

YASKAWA ELECTRIC (CHINA) CO., LTD.
12F, Carlton Bld., No.21 HuangHe Road, HuangPu District, Shanghai 200003, China
Phone 86-21-5385-2200 Fax 86-21-5385-3299

YASKAWA ELECTRIC (SHANGHAI) CO., LTD. BEIJING OFFICE
Room 1011, Tower W3 Oriental Plaza, No.1 East Chang An Ave.,
Dong Cheng District, Beijing 100738, China
Phone 86-10-8518-4086 Fax 86-10-8518-4082

YASKAWA ELECTRIC TAIWAN CORPORATION
9F, 16, Nanking E. Rd., Sec. 3, Taipei 104, Taiwan
Phone 886-2-2502-5003 Fax 886-2-2505-1280

YASKAWA ELECTRIC CORPORATION

In the event that the end user of this product is to be the military and said product is to be employed in any weapons systems or the manufacture thereof, the export will fall under the relevant regulations as stipulated in the Foreign Exchange and Foreign Trade Regulations. Therefore, be sure to follow all procedures and submit all relevant documentation according to any and all rules, regulations and laws that may apply.

Specifications are subject to change without notice for ongoing product modifications and improvements.

© 2008-2012 YASKAWA ELECTRIC CORPORATION. All rights reserved.

MANUAL NO. SIEP C880700 38B

Published in Japan July 2012 08-9 ◁1▷-5
12-6-6